brak
str. 13-16

mikroKLAN 18

6 1985

P. 1877/85

informatyka

Rachunki numeryczne w kodzie maszynowym

Pakiet SIMIS dla MERY 400

Język programowania C

Nauczyciel — uczeń — mikrokomputer

WYDAWNICTWO NOT

Cena 100 zł

informatyka

6 1985

Nr 6
Miesięcznik Rok XX
Czerwiec 1985

Organ Komitetu Informatyki MNSZWiT oraz Komitetu Naukowo-Technicznego NOT ds. Informatyki

KOLEGIUM REDAKCYJNE:

Dr inż. Wacław ISZKOWSKI, mgr Teresa JABŁOŃSKA (sekretarz redakcji), Władysław KLEPACZ (zastępca redaktora naczelnego), prof. dr hab. Leon ŁUKASZEWICZ (redaktor naczelny), mgr inż. Andrzej J. PIOTROWSKI, dr inż. Janusz ZALEWSKI

STALE WSPÓŁPRACUJĄ:

Mgr inż. Witold ABRAMOWICZ (Szwajcaria), mgr inż. Ryszard K. KOTT (Wielka Brytania), dr Jacek OWCZARCZYK, dr Andrzej SZAŁAS, mgr inż. Teresa WILCZEK

PRZEWODNICZĄCY RADY PROGRAMOWEJ:

Prof. dr hab. Tadeusz PECHE

Materiałów nie zamówionych redakcja nie zwraca

Redakcja: 00-041 Warszawa, ul. Jasna 14/16, pok. 243 i 244, tel. 27-71-40 lub 26-82-61 w. 184

Zakł. Graf. „Tamka". Zam. 0438-1300/85. Obj. 4,0 ark. druk. Nakład 6100 egz. N-10.

ISSN 0542-9951, INDEKS 36124

Cena egzemplarza zł 100,—
Prenumerata roczna zł 1200,—

WYDAWNICTWO NACZELNA ORGANIZACJA TECHNICZNA
SIGMA
CZASOPISM I KSIĄŻEK TECHNICZNYCH
00-950 Warszawa
skrytka pocztowa 1004
ul. Biała 4

W NUMERZE: Strona

Komputery SINCLAIRA — rachunki numeryczne w kodzie maszynowym
Jerzy Karczmarczuk 1

Pakiet SIMIS dla MERY 400
Marian Furman, Zbigniew M. Nowicki, Jerzy Solak 4

Język programowania C (3)
Julian Winiewski 7

mikroKLAN 11

Jak rozszerzyć BASIC ZX SPECTRUM

Doświadczenia z eksploatacji IMZ-80

Schemat ZX SPECTRUM

Polska Federacja Klubów Mikroinformatycznych

Akademia mikroKLANU (7). ACKERMAN

Test numeryczny

DYDAKTYKA

Nauczyciel — uczeń — mikrokomputer
Emanuel Czyżo 19

SAMOTESTY

III/B. Metody sortowania wewnętrznego 21

Z KRAJU 23

Mikrokomputery w Pałacu Młodzieży

Rozproszone systemy przetwarzania danych

ZE ŚWIATA 25

Dyski laserowe

Dyski 3M

TERMINOLOGIA 26

Problemy tworzenia terminologii

W NAJBLIŻSZYCH NUMERACH:

- Jacek Irlik o kwalifikacji prawnej systemu komputerowego
- Jan Bielecki o definiowaniu kompilatorów i instrukcji strukturalnych w języku FORTH
- Zbigniew Suraj o problemach nauczania informatyki w szkołach średnich
- Stanisław Waligórski o języku LOGO
- Wacław Iszkowski i Paweł Grzegorzewicz o systemie generowania systemów operacyjnych LOGOS
- Peter Broczko o produkcji mikrokomputerów w krajach RWPG
- Cezary Stępień i Jarosław Szyrkowiec o wielopoziomowym systemie oprogramowania pamięci kasetowych PK-1

JERZY KARCZMARCZUK

Instytut Fizyki
Uniwersytet Jagielloński
Kraków

P.1877/85

Komputery SINCLAIRA — rachunki numeryczne w kodzie maszynowym

W przypadku skomplikowanych problemów, dla których czas obliczeń jest parametrem krytycznym, mikrokomputery ZX81 i ZX SPECTRUM nie są zbyt użyteczne, jednak nawet mały ZX81 może być niekiedy przyzwoitym narzędziem obliczeniowym, np. dla fizyka — jeśli ominie się czasową nieefektywność BASICA i skorzysta bezpośrednio z wbudowanych procedur arytmetycznych. Artykuł ten jest przeznaczony dla użytkowników ZX81 i ZX SPECTRUM, którzy usiłują wykorzystać je do celów profesjonalnych i zależy im na przyspieszeniu obliczeń.

PROGRAMOWANIE W KODZIE MASZYNOWYM

Pisanie dużych programów wyłącznie w kodzie maszynowym jest kłopotliwe i nieopłacalne, gdyż strata ludzkiego czasu może znacznie przewyższyć oszczędność czasu maszyny. Takie czynności, jak wprowadzanie i drukowanie danych, rezerwacja miejsca na tablice liczbowe itp. — najlepiej wykonywać metodą najprostszą, tj. w BASICU, co ma dodatkową zaletę łatwego przerywania programu. Nieopłacalna jest konwersja na kod tych fragmentów programu, które stanowią jedno (nawet duże) wyrażenie arytmetyczne zawierające dużo stałych liczbowych. Stałe liczbowe w programie są kompilowane przez BASIC SINCLAIRA w momencie wprowadzania programu. Operowanie stałymi liczbowymi w kodzie nie jest najwygodniejsze i zabiera sporo czasu.

Największy zysk czasowy jest związany z możliwością wykonywania w kodzie szybkich operacji na małych liczbach całkowitych, mieszczących się w sprzętowych rejestrach Z80, a także — z możliwością ominięcia organizacyjnych nieefektywności BASICU, tj. liniowego przeszukiwania obszaru zmiennych przy każdym odwołaniu się do zmiennej oraz liniowego przeszukiwania programu przy każdym skoku GOTO, GOSUB czy NEXT.

Programowanie obliczeń w arytmetyce zmiennopozycyjnej w kodzie na ZX81I i ZX SPECTRUM (w tym zakresie oba komputery są prawie identycznie oprogramowane) jest złożone, ale nie jest trudne i może być ciekawe. Ponieważ interpreter BASICA, oprócz operacji jawnie nakazanych przez program, wykonuje szereg obliczeń wewnętrznych; np. wywołanie RND uruchamia podprogram realizujący algorytm:

```
SEED : = ((SEED + 1) * 75) mod 65537 — 1
RND  : = SEED/65536
```

— projektanci oprogramowania ZX, aby ułatwić sobie pracę, wbudowali w interpreter niejako maszynę w maszynie. Jest to wirtualny „kalkulator" ze stosem, mający własny język z rozkazami wykonującymi operacje arytmetyczne (wywołujące funkcje wbudowane), ale także rozkazy skoków i rozkazy ułatwiające konstrukcję własnych funkcji zadanych przez przybliżenia wielomianowe.

Kalkulator operuje na kilku specyficznych strukturach danych, wymienionych poniżej.

• **Stos kalkulatora.** Jego adres początkowy mieści się w zmiennej systemowej STKBOT, a koniec, tj. adres pierwszej wolnej komórki nad stosem — w zmiennej STKEND. Stos ten jest zorganizowany jako zwiększająca się w górę przestrzeni adresowej tablica 5-bajtowych elementów zawierających liczby zmiennopozycyjne lub deskryptory ciągów znaków (napisów, ang. string). Format liczb zmiennopozycyjnych jest podawany w podręcznikach, warto jednak przypomnieć, że pierwszy bajt zawiera cechę przesuniętą o 128, tj. dla cechy równej 128, wartość liczby pokrywa się z wartością mantysy mieszczącej się w czterech dalszych bajtach i ograniczonej do zakresu [1/2, 1] dla liczb dodatnich. Pierwszy bit znormalizowanej mantysy jest z definicji równy 1. W formacie przyjętym m.in. przez Sinclaira, bit ten jest wykorzystany do pamiętania znaku liczby — jeśli wynosi 0, liczba jest dodatnia (oczywiście podczas wykonywania operacji bit ten jest wstawiany), jeśli zaś 1, to liczba jest ujemna. Deskryptor napisu ma następujący format: pierwszy bajt informuje, czy jest to napis prosty czy tablica, dwa następne zawierają adres początku napisu (lub odpowiedniego fragmentu), dwa ostatnie — długość.

• **Systemowa tablica stałych:** 0, 1, 1/2, PI/2 oraz 10. Adres tej tablicy w ZX81 jest równy 1915, a w SPECTRUM — 32c5 (adresy i wszystkie liczby będą dalej podawane w kodzie szesnastkowym, chyba że zostanie zaznaczone inaczej). Format liczb w tej tablicy, a także w innych stałych, o których będzie mowa, jest niestandardowy!

• **Tablica pomocnicza.** Jej adres początkowy jest zawarty w zmiennej systemowej MEM. W większości wypadków zawartość MEM pokrywa się z adresem MEMBOT — 30-bitowej tablicy systemowej, jednak użytkownik może przypisać zmiennej MEM adres własnej tablicy, np. tablicy utworzonej przez program w BASICU, zapewniając w ten sposób komunikację między kalkulatorem a programem w BASICU.

• **Stałe liczbowe.** Są one umieszczone wewnątrz programu kalkulatora, np. jako współczynniki wielomianów.

• **Zmienna systemowa BREG** (oznaczana też BERG). Jest wykorzystywana jako pomocniczy rejestr do konstrukcji pętli w kodzie kalkulatora.

Kalkulator uruchamia się rozkazem procesora Z80, RST 28 (kod EF), który przekazuje sterowanie głównej pętli kalkulatora mieszczącej się pod adresem 199D (ZX81) oraz 335B (SPECTRUM). Przed wywołaniem kalkulatora zaleca się przechowanie zawartości rejestru HL' i odtworzenie go przed powrotem do BASICA. Należy również odtworzyć poprzednie wartości w przypadku przedefiniowania adresu stosu lub wartości MEM.

Po rozkazie RST 28 pisze się rozkazy w kodzie kalkulatora. Nie są one rozpoznawane przez asembler Z80, więc trzeba je wprowadzić numerycznie. Interpreter BASICA dość intensywnie wykorzystuje podprogramy kalkulatora, co powoduje, że mechaniczna deasemblacja pamięci ROM przez nieświadomych programistów dostarcza niekiedy wydruków bez sensu. Powrót do normalnego trybu, tj. kodu Z80, następuje po rozkazie kalkulatora ENDC (kod 34 dla

ZX81, 38 dla SPECTRUM). Kody kalkulatora są 1-bajtowe, nie licząc przesunięć dotyczących rozkazów skoku oraz wbudowanych w program stałych liczbowych i są traktowane (po ewentualnym przetworzeniu) jako indeksy w tablicy procedur wykonawczych. Adres tej tablicy wynosi 1923 (ZX81) i 32D7 (SPECTRUM).

Nieco inna mnemonika niż użyta w tym artykule, występuje w książkach Iana Logana pt. ZX81 i ZX SPECTRUM ROM Disassembly, wydawanych od 1981 roku przez Melbourne House.

Przez WW oznaczono liczbę lub deskryptor napisu znajdujący się na szczycie stosu (o adresie równym wartości STKEND-5), a przez DD — element stosu leżący pod WW. Dla operacji dwuargumentowych (o kodach 00-17) DD i WW są traktowane jako argumenty, a wynik jest wpisywany jako nowe WW w miejscu poprzedniego DD, ze zmniejszeniem STKEND o 5. Rozkazy o kodach większych niż 17 są jednoargumentowe i wymieniają poprzednie WW na nowe nie zmieniając wartości STKEND, względnie dokładają na stos nową liczbę z tablicy stałych, z programu lub z tablicy MEM.

KOD	ZX81	SPECTRUM	Opis dla SEPCTRUM!
00	JRTR	JRTR	skok względny warunkowy, jeśli WW≠0
01	EXCH	EXCH	zamiana wartości WW z DD
02	DELE	DELE	usunięcia WW; poprzednie DD staje się WW
03	—	—	odejmowanie WW od DD
02	*	*	mnożenie DD przez WW
05	/	/	dzielenie DD przez WW
06	!	!	podnoszenie DD do potęgi WW
07	OR	OR	logiczne „lub"; wynik jest równy 1, o ile WW≠0, w przeciwnym wypadku — DD
08	N&N	N&N	logiczne „i" dla liczb
09	N<=N	N<=N	DD mniejsze lub równe WW liczbowo? Wynik jest równy 1 jeśli prawda, w przeciwnym wypadku 0
0A	N>=N	N>=N	DD większe lub równe WW?
0B	N<>N	N<>N	DD nierówne WW?
0C	N>N	N>N	DD większe od WW?
0D	N<N	N<N	DD mniejsze od WW?
0E	N=N	N=N	DD równe WW?
0F	N+N	N+N	dodawanie liczb DD i WW
10	S&N	S&N	logiczne „i"; wynikiem jest napis w DD jeśli liczba WW≠0, w przeciwnym wypadku — pusty
11	S<=S	S<=S	Napisy w DD i WW są równe lub
12	S>=S	S>=S	DD poprzedza leksykalnie WW jw., lecz DD następuje po WW
13	S<>S	S<>S	Napisy w DD i WW są różne? Wynik jest równy 1 lub 0
14	S>S	S>S	DD leksykalnie następuje po WW
15	S<S	S<S	DD leksykalnie poprzedza WW
16	S=S	S=S	czy DD i WW są identycznymi napisami?
17	S+S	S+S	konkatenacja napisów — WW dołącza się do DD
18	NEG	WAL$	wywołanie funkcji wbudowanej
19	CODE	USR$	jw., wywołanie USR z argumentem znakowym (do UDG)
1A	VAL	RDIN	czytanie znaku z różnych strumieni
1B	LEN	NEG	zmiana znaku liczby na stosie
1C	SIN	CODE	wywołanie funkcji wbudowanej
1D	COS	VAL	jw.
1E	TAN	LEN	jw.
1F	ASN	SIN	jw.
20	ACS	COS	jw.
21	ATN	TAN	jw.
22	LN	ASN	jw.
23	EXP	ACS	jw.
24	INT	ATN	jw.
25	SQR	LN	jw.
26	SGN	EXP	jw.
27	ABS	INT	jw.
28	PEEK	SQR	jw.
29	USR	SGN	jw.
2A	STR$	ABS	jw.
2B	CHR$	PEEK	jw.
2C	NOT	IN	jw.
2D	DUPL	USRN	wywołanie USR z argumentem liczbowym
2E	MODQ	STR$	wywołanie funkcji wbudowanej
2F	JUMP	CHR$	jw.
30	LDD	NOT	jw.
31	DJNZ	DUPL	powielenie WW i położenie na stos jako nowe WW
32	0	MODQ	dzielenie całkowite DD przez WW; wynik jest umieszczony w WW, a reszta w DD
33	0	JUMP	bezwarunkowy skok względny
34	ENDC	LDD	załadowanie na stos stałej użytkownika
35	TRIG	DJNZ	zmniejszenie BRG o 1 i skok, jeśli BREG≠0
36	TRUN	0	jednoargumentowe porównanie: czy WW jest mniejsze od 0?
37	OPER	0	WW większe od zera?
38	EFMT	ENDC	rozkaz kończący sekwencję kodu kalkulatora
39	SERI	TRIG	przygotowanie argumentu dla funkcji SIN i COS
3A	LDC	TRUN	obcięcie WW do najbliższej liczby całkowitej
3B	STM	OPER	dynamiczna operacja, kod przekazany w rejestrze B
3C	LDM	EFMT	konwersja z formatu xEm
3D		IFMT	konwersja z formatu całkowitego
3E		SERI	obliczenie sumy rozwinięcia Czebyszewa
3F		LDC	ładowanie na stos stałej systemowej
40		STM	przesłanie WW do tablicy MEM
41		LDM	ładowanie na stos danej z MEM

PRZYKŁADOWE PROGRAMY

Przedstawiam tu przykład prostego programu obliczającego pierwiastek kwadratowy metodą Newtona. Metoda ta jest znacznie szybsza niż używana przez funkcję SQR, gdzie $x=\sqrt{y}$ oblicza się jako **x=exp (1/2*ln(ny))**. Podane kody dotyczą SPECTRUM. Aby przystosować ten fragment programu do ZX81 wystarczy zmienić kody rozkazów DUPL, NOT, JUMP i ENDC. Założono, że wartość y jest umieszczona w pierwszym elemencie tablicy MEM, a wynik — w drugim.

	Rozkaz	Kod	
	RST 28	EF	wywołanie kalkulatora
	LDC 1	A1	początkowa wartość x: 1
et	DUPL	31	na stosie: x, x
	DUPL	31	x, x, x
	LDM 0	E0	x, x, x, y
	EXCH	01	x, x, y, x
	/	05	x, x, y/x
	N+N	0F	x, x+y/x
	LDC 2	A2	x x+y/x,1/2
	*	04	x, (x+y/x)/2
	STM 1	C1	przesłanie wyniku do MEM
	—	03	x — (x+y/x)/2
	NOT	30	jeśli wynik zerowy: WW:=1
	JRTR kon	00 04	skończ i zakończ obliczenia
	LDM 1	E1	w przeciwnym wypadku pobierz nowe x
	JUMP et	2F F1	i kontynuuj iterację
kon	ENDC	38	koniec pracy kalkulatora

Program wraz ze wstępem i zakończeniem, których celem było przekazanie danych z programu i do programu w BASICU, wykonywał się tysiąc razy w pętli na SPECTRUM przez ok. 50 sekund. Dla porównania, pętla wywołująca SQR trwa 114 sekund, a tak samo zorganizowana pętla „pusta", zawierająca wywołanie nie wykonujące żadnego podprogramu — 14 sekund. Oszczędność byłaby jeszcze większa, gdyby zamiast dzielić przez dwa w kodzie kalkulatora, po prostu zmniejszyć wykładnik liczby o 1 w kodzie Z80.

Tablica rozkazów kalkulatora oraz przykład wymagają dokładniejszego komentarza.

Skok pusty jest to skok z przesunięciem 1, a nie 0, jak dla kodu Z80! Rozkaz JRTR usuwa WW ze stosu, a JUMP i DJNZ nie naruszają stosu.

Rozkazy SERI, LDC, STM i LDM w rzeczywistości są kodowane inaczej niż podano w tabeli, której kody stanowią bezpośrednie indeksy tablicy adresów. W 1-bajtowy kod rozkazu wbudowany jest parametr, np. dla obu komputerów rozkazy SERI mają kody 86, 88 i 8C (takich używa interpreter BASICA), odpowiadające wielomianom 6, 8 i 12 stopnia. Użytkownik może skonstruować własne rozkazy SERI — do 32 stopnia. Rozkazy LDC występujące w przykładzie, o kodach A0, A1,...,A4, ładują na stos odpowiednio: 0, 1, 1/2, PI/2 oraz 10 (dziesiętnie). Rozkazy STM mają kody C0, C1 itd., a LDM — kody E0, E1 itd. Maksymalny rozmiar tablicy MEM wynosi 32. W przypadku większych tablic należy podczas pracy z kalkulatorem przesuwać zawartość MEM.

Rozkaz DJNZ jest analogiczny do odpowiedniego rozkazu mikroprocesora Z80. Jako rejestr sterujący pętlą wykorzystywana jest zmienna BREG, do której w momencie wywołania kalkulator kopiuje zawartość rejestru B. Jeśli chcemy, aby kalkulator nie zmieniał wartości BREG, należy go wywołać przez CALL 19A4 dla ZX81 lub CALL 3362 dla SPECTRUM, zamiast przez RST 28.

Program kalkulatora można zakończyć rozkazem ENDC, wykonać fragment programu w kodzie Z80 i wrócić do kalkulatora nie tracąc zawartości stosu ani tablicy MEM. Takie rozłączne fragmenty kodu kalkulatora mogą komunikować się przez skoki; byle tylko wskutek błędu nie wykonać skoku do kodu Z80.

Rozkaz OPER jest często wykorzystywany przez interpreter BASICA. Pozwala on wykonać operację, której kod jest dynamicznie przekazany kalkulatorowi przez rejestr B.

Rozkaz IFMT dla SPECTRUM dokonuje konwersji WW z dopuszczalnego na tym komputerze formatu całkowitego na zmiennopozycyjny.

Rozkaz EFMT służy do wprowadzania liczb w formacie z potęgą dziesiątki, znanym z FORTRANU jako format E. Fragment zapisu liczby przed literą E znajduje się na stosie, a potęga dziesiątki — w rejestrze A. Tę operację można wykorzystać do mnożenia liczb przez potęgę dziesiątki.

Rozkaz TRIG zmienia argument na niewielką liczbę, dla której wielomianowe rozwinięcie funkcji trygonometrycznych jest już w miarę dokładne.

Program w kodzie kalkulatora może zawierać wbudowane stałe liczbowe. Rozkaz LDD powoduje, że następujący po nim ciąg bajtów jest traktowany jako stała liczbowa w specjalnym, skondensowanym formacie. Jeżeli **N0, N1**, itd. — oznacza kolejne bajty po rozkazie LDD, to zapis **N0 div 40+1** wyznacza, ile następnych bajtów (od 1 do 4) należy do stałej. Te bajty tworzą mantysę liczby, ewentualnie uzupełnioną zerami z prawej strony, a wykładnik jest dany przez **N0 mod 40+50**, o ile **NO mod 40** jest różne od zera. W przeciwnym wypadku wykładnikiem jest **N1+50**. Należy pamiętać, że jest to wykładnik przesunięty. Tak więc dla ZX81 skondensowana postać liczby dziesięć jest równa 34 20.

Jako przykład zapiszmy fragment programu obliczającego wartość funkcji Gamma Eulera $\Gamma(x+1)$ dla **x** od 0 do 1 przez przybliżenie wielomianem piątego stopnia (schematem Hornera). Wartość **x** jest podana w komórce MEM 0, a wynik — udostępniany w MEM 1.

Rozkaz	Kod dla SPECTRUM	
LD B,05	06 05	
REST 28	EF	
LDC 1	A1	jedynka: zerowy wyraz
LDD	34 F0 93 2A 53 90	—0.5748646
LDD	34 F0 73 84 38 DE	0.9512363
LDD	34 F0 B3 29 F2 42	—0.6998588
LDD	34 EF 59 5F 42 85	0.4245549
LDD	34 ED CE FC A2 7C	—0.1010678
LDM 0	E0	wartość zmiennej
*	04	
N+N	0F	
DJNZ et	35 FC	5 przebiegów pętli
STM 1	C1	
ENDC	38	

Użycie kalkulatora wyłącznie w celu obliczenia wartości takiego wyrażenia nie jest opłacalne, nawet jeśli będzie ono obliczane wiele razy, gdyż zysk czasowy (dzięki ominięciu analizatora wyrażeń w BASICU) jest zredukowany przez przetwarzanie skondensowanych stałych. Wygodniejsze dla człowieka i nieco szybsze dla maszyny będzie przekazanie stałych w tablicy MEM.

W obliczeniach funkcji elementarnych wbudowanych w BASIC nie korzysta się z rozwinięć Taylora, lecz z rozwinięć na wielomiany Czebyszewa. Sumowanie tych rozwinięć wykonuje się rozkazami SERI. Za rozkazem SERI znajduje się odpowiednia liczba stałych w formacie skondensowanym, traktowanych jako współczynniki rozwinięcia. Argument rozwinięcia znajduje się na stosie. Zagadnienie konstrukcji rozwinięć na wielomiany Czebyszewa jest jednak zbyt skomplikowane, aby je szczegółowo wyjaśnić w tym artykule.

Jak przekazywać kalkulatorowi dane z programu w BASICU i jak odbierać wyniki? Operowanie małymi liczbami całkowitymi jest niewygodne, lecz proste, gdyż można użyć instrukcji POKE wpisania do określonej komórki lub RANDOMIZE — umieszczającej wartość argumentu w SEED. Podprogram w kodzie może tę liczbę pobrać do rejestru BC i wywołać procedurę o adresie 1520 (ZX81) względnie 2D2B (SPECTRUM), która umieszcza wartość BC na stosie w formacie zmiennopozycyjnym. Odwrotną operacją, tj. przesłanie WW do BC, wykonuje procedura o adresie 158A (ZX81) lub 2DA2 (SPECTRUM). Należy przypomnieć, że USR udostępnia programowi w BASICU wartość rejestru BC.

Przesłanie liczb zmiennopozycyjnych jest bardziej skomplikowane. Jeden sposób polega na przekazaniu programowi w kodzie maszynowym nazwy tablicy (razem z nawiasem otwierającym), którą można przygotować gdziekolwiek w pamięci (chodzi o nazwę nie o tablicę). Adres nazwy należy umieścić w zmiennej systemowej CH_ADD i wywołać procedurę o adresie 111C (ZX81) lub 28B2 (SPECTRUM), która w rejestrze HL dostarczy adresu, pod którym umieszczony jest ostatni (dla tablic — jedyny) znak nazwy zmiennej w obszarze zmiennych. Następnie należy ominąć nagłówek zawierający informację o wymiarach tablicy i odpowiedni adres początku danych przekazać do MEM. Prostszym sposobem jest wykorzystanie zmiennej systemowej DEST. Przykładowo — jeżeli tablica M zawiera interesujące nas dane, to wywołanie podprogramu w kodzie przez instrukcję:

```
LET M (1) = USR ...
```

spowoduje, że do zmiennej DEST interpreter instrukcji LET przekaże adres o 1 mniejszy niż adres M(1) i wtedy dopiero nastąpi wywołanie USR. Dalsze działanie przebiega podobnie jak w metodzie pierwszej, przy czym dla tablic znakowych obowiązuje nieco inna konwencja.

SPECTRUM ma jeszcze jedną metodę, wygodną przy definiowaniu własnych funkcji w kodzie, które można następnie wywołać przez FN. Jeśli zdefiniujemy np.

```
DEF FN f (x) = USR ...
```

to w momencie wprowadzania tej definicji interpreter zarezerwuje w programie za literą **x** miejsce na wartość argumentu — 5 bajtów poprzedzone bajtem znacznika, identycznym jak dla wszystkich stałych liczbowych w programie. W momencie wywołania **FN f**, zmienna systemowa DEFADD będzie zawierać adres litery **x**, o 2 mniejszy niż adres wartości argumentu. Wartość tę można pobrać, można również zmienić i przekazać z powrotem przy użyciu niezbyt czystego chwytu:

```
DEF FN f (x) = USR (...) + x
```

Podprogram w kodzie pobiera argument, oblicza wynik, umieszcza w miejsce argumentu, a przed powrotem do BASICA zeruje rejestr BC. Interpreter „nie zauważy", że do zera doda już zupełnie inną wartość **x**. Oczywiście, to **x** nie ma nic wspólnego z ewentualną zmienną **x**.

* * *

Podana informacja o kalkulatorze ZX jest z konieczności wyrywkowa i bardzo skondensowana. Nie są to gotowe przepisy, lecz raczej materiał do dalszej samodzielnej pracy. Jej wyniki mogą się jednak bardzo opłacić. Zainteresowanych Czytelników proszę o listy (ul. Łokietka 19/90; 30-016 Kraków).

MARIAN FURMAN
ZBIGNIEW M. NOWICKI
JERZY SOLAK

Instytut Informacji Naukowej,
Technicznej i Ekonomicznej
Warszawa

Pakiet SIMIS dla MERY 400

Celem artykułu jest przedstawienie możliwości pakietu SIMIS dla minikomputera MERA 400. Pakiet ten został opracowany w Instytucie INTE w Warszawie i stanowi kontynuację wcześniej zrealizowanych w Instytucie przez ten sam zespół pakietów SIM-400 [4], SIM-400 M [1] i SIM-400 R [3]. Ze względu na swą uniwersalność pakiety te znalazły szerokie zastosowanie wśród użytkowników maszyn MERA 400 przy realizacji systemów wyszukiwania informacji. Ze względu na istniejące ograniczenia w zakresie pojemności pamięci przyjęto w nich prostą strukturę zbiorów danych o dostępie sekwencyjnym. Dzięki temu pakiety serii SIM-400 można było z powodzeniem wykorzystywać do tworzenia i przetwarzania zbiorów danych o niezbyt wielkich rozmiarach (rzędu kilku tysięcy dokumentów). Kolejne pakiety tej serii były tworzone jako rozszerzenia możliwości wersji poprzednich. I tak, SJM-400 M dzięki wprowadzeniu pól numerycznych pozwalał łączyć funkcje informacyjno-wyszukiwawcze SIM-400 z funkcjami typowymi dla systemów przetwarzania danych, natomiast SIM-400 R rozwijał możliwości wyszukiwawcze swoich poprzedników, poprzez stworzenie możliwości badania relacji i odwoływania się do wyników wyszukiwania przy wykonywaniu innych operacji. W skład pakietu SIM-400 R wchodzi konwersacyjny moduł wyszukiwania, natomiast wszystkie pozostałe funkcje są realizowane w sposób wsadowy.

Pakiet SIMIS zachowuje wszystkie możliwości SIM-400 R, najbardziej rozwiniętego pakietu serii SIM-400. Aparat wyszukiwawczy pakietu SIMIS został wzbogacony o możliwości równoczesnego odwoływania się w zapytaniu wyszukiwawczym do dwóch zbiorów danych. Stwarza to jakościowo nowe możliwości, między innymi umożliwia prostą realizację operacji typowych dla systemów zarządzania relacyjnymi bazami danych takich, jak selekcja, projekcja i złączenie. Ponadto SIMIS został wyposażony w elastyczny aparat operowania wieloma zbiorami danych, identyfikowanymi poprzez nazwy. Możliwości te oraz przeniesiony z poprzednich pakietów tej serii aparat obsługi zapytań przedstawionych jako złożone wyrażenia logiczne sprawiają, że SIMIS łączy w sobie cechy właściwe typowym systemom informacyjno-wyszukiwawczym oraz systemom zarządzania bazami danych.

OGÓLNA CHARAKTERYSTYKA PAKIETU

Baza danych SIMIS składa się ze zbiorów danych oznaczonych unikalnymi nazwami. Zbiór danych obejmuje dokumenty jednego rodzaju. Każdy dokument podzielony jest na pola opatrzone unikalnymi (w skali danego zbioru) etykietami. Wszystkie dokumenty wchodzące w skład jednego zbioru mają taką sam strukturę. Strukturę dokumentu definiuje się w trakcie zakładania bazy danych, określając liczbę pól, ich etykiety oraz typy. Dopuszczalne są pola dwóch typów:
— znakowe, które mogą zawierać dowolne ciągi znaków,
— liczbowe, z których każde zawiera pojedynczą liczbę.
Długość dokumentu nie jest stała i zależy od jego zawartości.

Pakiet SIMIS został zrealizowany jako zespół współpracujących ze sobą modułów programowych. Przeznaczenie poszczególnych modułów jest następujące:

EDYTOR — zakładanie zbiorów danych i ich aktualizacja,

SCALAJ — łączenie w jeden zbiór dwóch posortowanych zbiorów o podanej strukturze dokumentów,

SŁOWNIK — sporządzenie alfabetycznego wykazu słów występujących we wskazanych polach dokumentów,

INDEKS — sporządzanie alfabetycznego wykazu deskryptorów użytych do opisu dokumentów; z każdym deskryptorem jest powiązana lista numerów odpowiednich dokumentów,

OBLICZ — wykonywanie operacji arytmetycznych na polach liczbowych dokumentów,

SIMIS — wyszukiwanie oraz przeglądanie, drukowanie i sortowanie zbiorów.

Pierwsze pięć modułów służy do wsadowego przetwarzania zbiorów danych. Ich funkcje oraz sposób sterowania w znacznej mierze są takie same jak w przypadku noszących identyczne nazwy modułów pakietu SIM-400 R. Nowe funkcje pakietu zostały zgrupowane w module SIMIS, który realizuje przetwarzanie danych w trybie konwersacyjnym. Wybór trybu pracy modułów był podyktowany charakterem realizowanych przez nie funkcji. Moduły realizujące funkcje wykonywane stosunkowo rzadko oraz takie, które nie wymagają bezpośredniego udziału użytkownika, zostały zrealizowane w postaci wsadowej. W module konwersacyjnym skupiono natomiast funkcje wykonywane często i te, w których wskazany jest bezpośredni udział użytkownika.

Rezultatem wyszukiwania realizowanego w module SIMIS jest utworzenie nowego zbioru dokumentów. Nazwę i strukturę tego zbioru, jak również jeden bądź dwa zbiory wejściowe oraz warunek wyszukiwawczy, określa użytkownik w trakcie formułowania zlecenia wyszukiwawczego. Zbiór wyszukanych dokumentów można wyprowadzić na konsolę użytkownika. Podczas przeglądania tego zbioru można ręcznie wskazywać te dokumenty, które zostaną przeniesione do innego, nowo tworzonego zbioru. Pozwala to w trakcie przeglądania realizować wyszukiwanie według kryterium, którego nie można sformalizować jako warunku wyszukiwawczego. Zbiory wyszukanych dokumentów są zbiorami roboczymi, co oznacza iż z chwilą zakończenia seansu przestają one być dostępne dla użytkownika. Zbiory te można „utrwalić" za pomocą odpowiedniej dyrektywy, co zapewnia ich późniejsze przetwarzanie. Przykładem pomocniczych funkcji modułu SIMIS jest tworzenie zbiorów trwałych oraz informowanie o zbiorach dostępnych i o strukturze dokumentów wskazanego zbioru.

Z punktu widzenia użytkownika najbardziej istotną funkcją pakietu SIMIS pest wyszukiwanie, które wymaga bliższego omówienia.

MOŻLIWOŚCI WYSZUKIWAWCZE PAKIETU

W celu zlecenia operacji wyszukiwawczej użytkownik wprowadza komendę UTWORZ. Przed przystąpieniem do wyszukiwania system żąda od użytkownika podania niezbędnych danych wejściowych, mianowcie:

— **określenie zbiorów wejściowych** — należy podać nazwy jednego lub dwóch zbiorów podlegających przeszukiwaniu;

— **określenia zbioru wynikowego** — należy podać nazwę zbioru roboczego, w którym zostaną umieszczone wyszukane dokumenty, a także strukturę dokumentów tego zbioru (liczbę pól oraz ich etykiety i typy) oraz sposób ich tworzenia. Dokument wynikowy jest budowany z pól doku-

mentu wejściowego (dokumentów wejściowych). Definiując pole dokumentu wynikowego użytkownik wskazuje jednocześnie to pole wejściowe, którego zawartość zostanie doń przeniesiona;

— **zapytania wyszukiwawczego** — ma ono postać wyrażenia logicznego zbudowanego z relacji (operandów), operatorów logicznych (AND &, OR #, NOT) oraz nawiasów. Każda z relacji zbudowana jest z etykiety pola wejściowego, operatora relacji i wzorca. Dozwolone są operatory standardowych relacji równości, mniejszości i większości oraz relacji złożonych z wymienionych relacji prostych (np. >=). Dla wygody użytkownika przyjęto, że relacje złożone mogą być oznaczane za pomocą jednego z dwóch równoważnych operatorów, i tak >= oraz >= oznaczają tę samą relację. Dopuszcza się użycie relacji występowania, oznaczanej operatorem = ?. Relacja ta jest spełniona wówczas, gdy we wskazanym polu występuje ciąg znaków pokrywający się z zadanym wzorcem i rozpoczynający się od początku słowa [1]). Użycie tej relacji pozwala zatem na wyszukiwanie (z maskowaniem symboli z prawej strony). Wszystkie wymienione wcześniej relacje dotyczą pełnej zawartości wskazanego pola.

W relacji jako wzorzec może zostać użyta etykieta pola wejściowego albo stała. Stałe znakowe ogranicza się symbolami gwiazdki. Interpretacja relacji zależy od typu badanego pola i wzorca. Jeśli pole i wzorzec są typu liczbowego, relacja ma sens arytmetyczny. Gdy natomiast bada się pole znakowe, to relacja ma sens leksykograficzny. Oczywiście, relacji występowania można użyć jedynie w odniesieniu do pola znakowego.

Sposób wykonania operacji wyszukiwania zależy od tego czy wskazano jeden czy dwa zbiory wejściowe. Jeśli wskazano jeden taki zbiór, to zadany warunek bada się kolejno dla poszczególnych dokumentów tego zbioru i na podstawie każdego dokumentu spełniającego to kryterium tworzy się jeden dokument wynikowy. Gdy natomiast wskazano dwa zbiory wejściowe, warunek bada się dla wszystkich par dokumentów wejściowych, pochodzących z różnych zbiorów, i dla każdej pary spełniającej ten warunek tworzy się jeden dokument wynikowy. Jeśli zatem wskazano dwa zbiory wejściowe, to liczebność zbioru wynikowego może osiągnąć wartość równą iloczynowi liczebności zbiorów wejściowych. W zbiorze wynikowym mogą wystąpić powtórzenia dokumentów. Po przejrzeniu wszystkich dokumentów wejściowych zbiór wynikowych dokumentów jest poddawany operacji eliminacji powtórzeń. Na zakończenie użytkownik jest informowany o liczebności utworzonego zbioru wynikowego.

TRASY

IDTRAS	— identyfikator trasy
TRASA	— lista odwiedzanych miejscowości
TRANS	— środek transportu
OPIS	— skrótowy opis wycieczki, zawierający m.in. określenie celu, proponowane rozrywki, zajęcia sportowe itp.
CENA	— bazowa cena wycieczki (pole numeryczne)

WYCIECZKI

IDWYC	— identyfikator wycieczki
IDTRAS	— identyfikator trasy (wskazujący odpowiedni rekord w zbiorze TRASY)
DATWYJ	— data wyjazdu
DATPOW	— data powrotu
ULGA	— zniżka względem ceny bazowej (pole numeryczne)

KLIENCI

NAZW	— nazwisko i imię klienta
ADRES	— pełny adres klienta
IDWYC	— identyfikator wycieczki, wskazujący odpowiedni rekord w zbiorze WYCIECZKI

Rys. 1. Przykładowa baza danych

[1]) Słowo definiuje się jako ciąg znaków złożony z liter i cyfr. Ogranicznikami słów są zatem symbole specjalne, w tym spacja. Dla umożliwienia wyszukiwania fraz zbudowanych z kilku słów, definiuje się specjalny typ pola, nazywany polem deskryptorowym. W takim polu ogranicznikiem słowa jest jedynie przecinek.

Dla zilustrowania możliwości wyszukiwawczych pakietu SIMIS zostanie przedstawionych kilka przykładów dyrektywy UTWÓRZ. Dyrektywa ta zapewnia realizację operacji typowych dla systemów zarządzania relacyjnymi bazami danych [2]. Wszystkie przykłady dotyczą działalności pewnego biura podróży, związanej z organizacją grupowych wyjazdów turystycznych [6]. Przykładowa baza danych biura podróży obejmuje trzy zbiory danych, których strukturę przedstawiono na rys. 1. Wszystkie pola, z wyjątkiem tych, które określono w sposób jawny jako numeryczne, są polami znakowymi. Zbiór TRASY zawiera ogólne opisy tras wyjazdowych proponowanych przez rozważane biuro, natomiast zbiór WYCIECZKI składa się z dokumentów opisujących wszystkie wycieczki, jakie zostaną zorganizowane na tych trasach w danym okresie. Dokumenty w zbiorze TRASY opatrzone są unikalnymi identyfikatorami (pole IDTRAS), to samo dotyczy dokumentów ze zbioru WYCIECZKI (identyfikator IDWYC). Każdy dokument zbioru WYCIECZKI jest powiązany z dokumentem zbioru TRASY, który zawiera ogólne dane dotyczące trasy wycieczki. Należy zwrócić uwagę, że kilka dokumentów z WYCIECZKI może odwoływać się do tego samego dokumentu z TRASY, mogą to być przykładowo wycieczki odbywane po tej samej trasie w różnych terminach. Trzeci zbiór danych KLIENCI zawiera dane dotyczące uczestników wycieczek organizowanych przez rozważane biuro podróży. Ze zrozumiałych względów dokumenty tego zbioru są powiązane z dokumentami wchodzącymi w skład bioru WYCIECZKI. Gdy ta sama osoba uczestniczy w kilku wycieczkach, w zbiorze KLIENCI tworzy się kilka dokumentów zawierających identyczne dane personalne.

```
a) UTWORZ
   zbiór wejściowy 1: TRASY
   zbiór wejściowy 2:
   zbiór wyjściowy: POCIĄGIEM—DO—PRAGI
   pola: 1
   warunek: TRASA = ? * PRAGA * & TRANS = * POCIĄG *

b) UTWORZ
   zbiór wejściowy 1: TRASY
   zbiór wejściowy 2:
   zbiór wyjściowy: TRASY—WYCIECZEK
   pola: IDTRAS, TRASA
   warunek:

c) UTWORZ
   zbiór wejściowy 1: TRASY
   zbiór wejściowy 2: WYCIECZKI
   zbiór wyjściowy: WYCIECZKI—DO—DREZNA
   pola: IDWYC, TRASA, CENA, ULGA, DATWYJ,
     DATPOW
   warunek:  1.IDTRAS = 2. IDTRAS & TRASA = ? * DREZ-
     NO *

d) UTWORZ
   zbiór wejściowy 1: WYCIECZKI—DO—DREZNA
   zbiór wejściowy 2: KLIENCI
   zbiór wyjściowy: TURYŚCI—DO—DREZNA
   pola: NAZW, ADRES, DATWYJ, DATPOW
   warunek:  1.IDWYC = 2. IDWYC & DATWYJ >= 840101
     & DATPOW <= 841231
```

Rys. 2. Przykłady dyrektyw UTWORZ

Przykłady dyrektyw UTWORZ odnoszących się do bazy danych (przedstawione na rys. 1) podano na rys. 2. Teksty wprowadzane przez użytkownika zostały wyróżnione przez użycie dużych liter. Pierwszy z przykładów (a) dotyczy typowej operacji wyboru (selekcji). Wyszukiwane są wszystkie trasy podróży, które obejmują Pragę, a środkiem transportu jest pociąg. Dla wykonania tej operacji potrzebny jest tylko jeden zbiór wejściowy (zbiór TRASY). W rozważanym przypadku warunek ma postać iloczynu logicznego dwóch relacji. Wyniki wyszukiwania umieszczone są w zbiorze POCIĄGIEM_DO_PRAGI. Dokumenty tego zbioru będą miały identyczną budowę jak dokumenty pierwszego zbioru wejściowego, czyli zbioru TRASY. Użytkownik podejmuje taką decyzję podając cyfrę 1 jako odpowiedź na zapytanie systemu dotyczącego pól zbioru wyjściowego.

Drugi przykład dyrektywy UTWORZ (b) dotyczy typowej operacji rzutowania (projekcji). Podobnie jak w poprzednim przykładzie, wykorzystywany jest tylko jeden zbiór wejściowy. W odróżnieniu od operacji wyboru, w przypadku rzu-

towania nie podaje się warunku wyszukiwawczego (ściśle, podaje się warunek pusty, który jest zawsze spełniony). Powoduje to, że dokumet wynikowy zostaje utworzony dla każdego dokumentu wejściowego. Dokument wynikowy nie zawiera wszystkich pól dokumentu wejściowego, ale jedynie wybrane z nich. W omawianym przykładzie do dokumentu wynikowego przenoszone są pola IDTRAS i TRASA, przy czym etykiety ich nie ulegają zmianie. Należy zwrócić uwagę, że chociaż zamierzeniem użytkownika jest utworzenie zbioru wynikowego zawierającego dane dotyczące wyłącznie przebiegu trasy, to włącza on do dokumentu wynikowego — oprócz pola TRASA — również pole IDTRAS, które — jak wspomniano — jest identyfikatorem dokumentów w zbiorze TRASY. Zamieszczenie tego pola zapobiega utożsamieniu w zbiorze wynikowym wszystkich dokumentów dotyczących tej samej trasy, ale pochodzących z różnych dokumentów wejściowych (opisujących podróże różniące się np. użytym środkiem transportu).

Oczywiście, opisane wyżej operacje selekcji i projekcji można połączyć. Wówczas w dyrektywie UTWORZ występuje zarówno lista pól wyjściowych jak i warunek wyszukiwawczy.

Trzeci przykład (c) dotyczy bardziej złożonego działania, które jest połączeniem dwóch typowych operacji dla systemów zarządzenia relacyjnymi bazami danych: złączenia i projekcji. To działanie wymaga wskazania dwóch zbiorów wejściowych, na podstawie których tworzony jest zbiór wynikowy. Niezbędne jest określenie warunku wyszukiwawczego, który zostanie zbadany dla każdej pary dokumentów wejściowych. W przykładzie warunek ten jest spełniony, gdy oba dokumenty wejściowe odnoszą się do tej samej trasy (tzn. pola IDTRAS mają tę samą wartość w obu dokumentach) oraz trasa ta obejmuje Drezno. Trzeba zwrócić uwagę, że pola wejściowe o tych samych etykietach są rozróżniane przez użycie etykiet kwalifikowanych, tj. etykiet poprzedzonych identyfikatorem zbioru wejściowego 1. lub 2.). W rozważanym przykładzie do dokumentu wynikowego są kopiowane pola TRASA i CENA, ze zbioru TRASY, oraz pola IDWYC, ULGA, DATWYJ i DATPOW — z drugiego zbioru wejściowego (WYCIECZKI).

Oczywiście warunek logiczny dotyczący dwóch zbiorów wejściowych może być bardziej złożony od rozważanego powyżej, co pozwala na obsługę bardziej skomplikowanych zapytań wyszukiwawczych. Na przykład, zastąpienie warunku podanego w punkcie c warunkiem:

(TRASA = ? * DREZNO * # TRASA = ? * LIPSK *) & 1. IDTRAS = 2. IDTRAS & DATWYJ = > 85061 & DATPOW < = 850930 & CENA < = 15000

spowoduje wyszukiwanie danych dotyczących wycieczek do Drezna lub Lipska latem 1985 roku, których cena nie przekracza 15 tys. zł. Należy zwrócić uwagę, że zastosowany sposób zapisu dat umożliwia ich bezpośrednie porównywanie (daty późniejsze są zarazem „większe" leksykograficznie).

Dla wykonania operacji angażującej więcej niż dwa zbiory wejściowe należy posłużyć się kilkoma dyrektywami UTWORZ, z których każda (z wyjątkiem pierwszej) używa jako zbioru wejściowego zbioru utworzonego w rezultacie wykonania jednej z poprzednich dyrektyw. Dla ilustracji tej techniki rozważone zostanie zadanie polegające na wyszukaniu danych o wszystkich klientach, którzy uczestniczyli w wycieczkach do Drezna w 1984 roku. Łatwo zauważyć, że dla realizacji tego zadania należy posłużyć się danymi z trzech zbiorów. Trzeba w tym celu użyć dwóch dyrektyw UTWORZ. Jednym z możliwych rozwiązań jest podanie dyrektyw przedstawionych w przykładzie pod literami c i d. Pierwsza z tych dyrektyw została uprzednio omówiona. Druga dyrektywa zapewnia powiązanie odpowiadających sobie dokumentów ze zbiorów WYCIECZKI DO _DREZNA i KLIENCI oraz wybranie tych, które dotyczą wycieczek odbytych w 1984 roku.

* * *

Z przedstawionego opisu wynika, że pakiet SIMIS dysponuje prostym i wygodnym w obsłudze aparatem wyszukiwawczym, zapewniającym realizację operacji typowych dla systemów zarządzania relacyjnymi bazami danych. Istnieje również możliwość formułowania zapytań wyszukiwawczych w postaci złożonych wyrażeń logicznych zawierających relacje, co jest cechą charakterystyczną typowych systemów informacyjno-wyszukiwawczych. SIMIS odznacza się również wieloma innymi możliwościami, istotnymi z punktu widzenia rozmaitych zastosowań. Dla przykładu można wymienić wykonywanie obliczeń z użyciem pól liczbowych. Użytkownik dysponuje bardzo wygodnym i silnym aparatem definiowania tych obliczeń w postaci programu złożonego z instrukcji podstawienia, warunku oraz skoku.

Przedstawione w artykule przykłady dotyczyły zastosowania pakietu SIMIS w działalności biura podróży. Zakres możliwości stosowania tego pakietu jest bardzo szeroki. Do podstawowego obszaru zastosowań należy działalność ośrodków informacji i bibliotek [5], jednakże pakiet SIMIS może być użyty z powodzeniem również do prowadzenia ewidencji kadr, wspomagania działalności niewielkich zespołów badawczych lub automatyzacji rachunkowości.

Dostępna aktualnie pierwsza wersja pakietu SIMIS korzysta ze standardowego oprogramowania obsługi zbiorów dyskowych (pakiet File Maintenance Control) w ramach systemu operacyjnego SOM-3. Przygotowanie dysków do użycia w SIMIS jest bardzo proste i polega na utworzeniu na pakiecie dyskowym sekcji oznaczonych ustalonym hasłem dostępu (standardowo jest to hasło SIMIS). Pakiet SIMIS poszukuje zbiorów danych w tych właśnie sekcjach i zapewnia dostęp do zbiorów o podanej nazwie. W ten sposób użytkownik został uwolniony od żmudnych czynności dołączania zbiorów danych do strumieni systemu operacyjnego, co zwykle było czynnością wymagającą znajomości tego systemu oraz źródłem wielu pomyłek.

Plany rozwojowe pakietu SIMIS obejmują m.in. włączenie bardziej rozbudowanej kontroli poprawności dokumentów wprowadzanych do bazy danych. Celowe wydaje się również zdefiniowanie nowego typu pola, tzw. pola atrybutowego, przeznaczonego do przechowywania pojedynczej wartości z określonego zbioru (przykładem pól tego typu są pola TRANS, IDTRAS i IDWYC oraz DATWYJ i DATPOW z rys. 1). Szczególnym przypadkiem pola atrybutowego byłoby pole numeryczne.

Realizacja planów rozwojowych pakietu SIMIS zależy w dużej mierze od zapotrzebowania ze strony jego użytkowników. Osoby zainteresowane wykorzystaniem tego pakietu mogą otrzymać wszelkie informacje na ten temat u autorów artykułu.

LITERATURA

[1] Bańkowski J. i inni: Uniwersalny pakiet przetwarzania i wyszukiwania informacji dla MERY 400. INFORMATYKA, 9, 1980, str. 15—17.

[2] Date C. J.: Wprowadzenie do baz danych. WNT, Warszawa, 1981.

[3] Furman M. i inni: Rozszerzenia pakietu informacyjno-wyszukiwawczego SIM-400M. INFORMATYKA, 5, 1983, str. 24—26.

[4] Nowicki Z.: Computerised information service: Minicomputer Application. Int. Forum Inf. Doc., 2, 1981, str. 25—27.

[5] Nowicki Z., Solak J.: Obsluživanie kompleksnych informacjonnych zaprosov v režimie on-lajn s ispolzowaniem mini-evm. Konf. Bazy danych v setiach evm. MCNTI, Moskwa 1984, str. 213—218.

[6] Nowicki Z., Solak J.: Office automation with the minicomputer system SIMIS. Proc. of Int. Symp. „Network in Office Automation", North-Holland (w druku).

Ogłoszenia • Ogłoszenia • Ogłoszenia • Ogłoszenia

Michał Choroszucha poszukuje MACRO-8000 oraz innego oprogramowania Z-8000. Warszawa, tel. 10-26-68.

EO/923/K/85

Ogłoszenia • Ogłoszenia • Ogłoszenia • Ogłoszenia

JULIAN WINIEWSKI

Instytut Podstaw Informatyki PAN
WARSZAWA

Język programowania C (3)

W dwóch pierwszych częściach przedstawiono definicję języka C. Poniżej omówiono natomiast otoczenie języka oraz przykłady programów.

Zarówno wejście-wyjście jak i inne środki komunikacji z otoczeniem nie są częścią definicji języka C lecz stanowią treść jego bibliotek standardowych. Biblioteka libc jest ładowana automatycznie wraz z programem przez kompilator C. Na bibliotekę tę składają się funkcje opisane w podręczniku systemu UNIX [1]). Inna biblioteka, libm zawiera funkcje matematyczne — np. trygonometryczne, pierwiastek, potęgę. Funkcje z tej biblioteki zostaną włączone do programu wynikowego, o ile program źródłowy jest skompilowany z odpowiednią opcją. Deklaracje dla tych funkcji znajdują się w pliku <math.h>, który musi być włączony do programu używającego funkcji matematycznych. Pewna grupa funkcji z biblioteki libc stanowi standardowy pakiet wejścia-wyjścia (stdio).

STANDARDOWA BIBLIOTEKA libc

Pakiet stdio zawiera zestaw funkcji i makrodefinicji służących do operowania strumieniami i stanowi środek do tzw. buforowego wejścia-wyjścia. Strumień jest to plik ze związanym z nim buforem, deklarowany jako wskaźnik do typu FILE. Utworzenie (otwarcie) strumienia jest wykonywane funkcją fopen, która udostępnia wskaźnik będący odtąd identyfikatorem strumienia w dalszych operacjach. Strumień otwarty uprzednio przez funkcję fopen jest zamykany funkcją fclose. Argumentem funkcji fopen jest nazwa pliku, dla którego otwiera się strumień, oraz rodzaj otwarcia, np. — zapis, odczyt. W pakiecie stdio istnieją trzy — zazwyczaj otwarte — strumienie o stałych wskaźnikach, związane z trzema otwartymi standardowymi plikami:

stdin — standardowy plik wejściowy,

stdout — standardowy plik wyjściowy,

stderr — standardowy plik wyjściowy dla błędów.

Dowolna funkcja standardowa działająca na strumieniu, jeśli napotka koniec pliku lub błąd udostępni wartość całkowitą EOF. Deklaracja typu FILE oraz definicja stałej EOF znajdują się w pliku <stdio.h> i dlatego każdy program korzystający z funkcji pakietu stdio musi włączać ten plik, tj. musi zawierać dyrektywę

```
#include <stdio.h>
```

Najprostszym środkiem wejścia-wyjścia jest makrodefinicja getc (putc) wprowadzająca (wyprowadzająca) jeden znak z podanego strumienia. Wywołania getchar i putchar działają tak samo, ale na strumieniach standardowych, tj. getchar () i getc (stdin) oraz putchar (c) i putc (c, stdout) są parami równoważne. Wywołanie fgetc (fputc) działa tak jak getc (putc), ale jest funkcją, nie makrodefinicją. Natomiast getw (putw) wczytuje słowo zamiast bajtu.

Funkcjami wyższego rzędu są np. gets i puts oraz fgets i fputs:

- gets (s) wczytuje tekst (ciąg znaków zakończony znakiem nowego wiersza) ze strumienia stdin i umieszcza pod adresem s,
- fgets (s, n strumień) wczytuje n—1 znaków (lub do znaku nowego wiersza) ze strumienia o podanej nazwie,
- puts (s) i fputs (s, strumień) wyprowadzają tekst spod adresu s na strumień odpowiednio stdout lub podany.

Bardzo ważną grupą są funkcje wejścia-wyjścia z konwersją na określony format. Funkcjami wyjścia są: printf — wyprowadzająca wynik na strumień stdout, fprintf — na dowolny strumień, a sprintf — pod podany adres (zmienną tekstową) dodając znak końca tekstu (NULL). Wywołanie funkcji printf:

```
printf ("%d/n",2*x)
```

spowoduje, że wartość zmiennej x pomnożona przez 2 zostanie wyprowadzona na plik stdout w postaci dziesiętnej całkowitoliczbowej, po czym nastąpi wyprowadzenie znaku nowego wiersza. Funkcje te mogą wyprowadzać wartości większej liczby wyrażeń. Ogólna postać wywołania tej funkcji jest następująca:

```
printf (tekst_sterujący[,wyr1,...,wyrn])
```

Każdemu wyrażeniu na liście parametrów odpowiada jeden format w tekście sterującym. Formaty w tekście sterującym zaczynają się od znaku procenta (%), a wszystkie pozostałe znaki są wyprowadzane dosłownie. Dozwolone są następujące formaty:

%d %o oraz %x — wynik jest przekształcany do liczby całkowitej i wyprowadzany w postaci odpowiednio dziesiętnej, ósemkowej lub szesnastkowej;

%c — wartość jest przekształcana na typ char;

%s — wartość jest wyprowadzana jako tekst;

%f — wartość jest przekształcana na typ float.

Ponadto każdy z formatów może zawierać liczbę określającą minimalną długość pola (liczbę znaków) wyprowadzanej wartości (np. %4d). Jeśli wartość ma długość mniejszą niż pole, to dopełnia się je spacjami z lewej strony. Zamiast spacji dla wartości liczbowych można wyprowadzać nieznaczące zera umieszczając w formacie znak 0 przed długością pola (np. %04d). Wartość może być w swym polu dosuwana do lewej strony, co określa znak minus (np. %—20s). Dla tekstów oraz wartości zmiennoprzecinkowych można podawać precyzję. Określa ją liczba poprzedzona kropką np. %.5f %—20.12s), oznaczająca liczbę wyprowadzanych znaków tekstu lub liczbę wyprowadzanych cyfr po przecinku w formacie zmiennoprzecinkowym. Przykładowo, jeżeli x ma wartość 25, a s — „abcdefghijkl", to wywołanie

```
printf ("x = %3d, s = %—5.2s a/n %03d, %6.4s b",x,*s,++x,s[2])
```

spowoduje wyprowadzenie napisu

```
x = 025, s = ab    a
    026,     cdef  b
```

Funkcje scanf, fscanf i sscanf są odpowiednikami powyższych, dotyczącymi wprowadzania. Ich argumentami są, oprócz tekstu sterującego, adresy pod które należy wprowadzić wczytane wartości po przekształceniu ich zgodnie z formatami. Przykładowo, dla deklaracji

```
int i; float x; char nazwa [30];
```

[1]) Omawianie UNIXA rozpoczniemy w następnym numerze INFORMATYKI (przyp. red.).

oraz tekstu wejściowego

```
25 54,32E—1 xxxx
```

wywołanie

```
scanf ("%d%f%s"&i,&x, nazwa);
```

przypisze zmiennej i wartość 25, zmiennej x wartość 5.432, a tekst „xxxx" zmiennej nazwa. Dla tych samych deklaracji wywołanie

```
scanf ("%2d %f %d %*d %2s",&i,&x, nazwa);
```

dla tekstu

```
56789 0123 45a78
```

przypisze zmiennej i wartość 56, zmiennej x wartość 789,0, następnie opuści liczbę całkowitą z tekstu wejściowego (0123), po czym zmiennej nazwa nada wartość „45". Następne wywołanie dowolnej funkcji wprowadzania danych z tego strumienia rozpocznie działanie od litery a w tym ciągu.

Inną funkcją tego pakietu jest ungetc, która umieszcza określony znak w określonym strumieniu wejściowym. Następne wywołanie getc udostępnia ten znak.

Przydatnymi funkcjami są fseek, ftell i rewind, działające na wskaźnik aktualnej pozycji w strumieniu:

- fseek (strumień, odl, baza) przesuwa ten wskaźnik na odległość odl bajtów od początku pliku (gdy baza = 0), od aktualnej pozycji (baza = 1), lub od końca pliku (baza = 2); odl jest liczbą całkowitą typu long ze znakiem, którą dodaje się do bazy,
- rewind (strumień) jest równoważne fseek (strumień, 0L, 0),
- ftell (strumień) udostępnia odległość aktualnej pozycji od początku pliku liczoną w bajtach.

Warto wymienić także funkcje dwójkowego wejścia-wyjścia fread i fwrite:

- fread (ptr, sizeof (*ptr), nb, strumień) wczytuje do bloku pamięci, zaczynającego się od adresu wskazywanego przez ptr, obiekty typu *ptr z określonego strumienia (ich ilczbę określa nb),
- fwrite dopisuje do określonego strumienia nb obiektów typu lub *ptr) spod adresu wskazywanego przez ptr.

Wszystkie opisane funkcje pakietu stdio odwołują się do funkcji systemowych, które należą również do biblioteki libc. Systemowe funkcje wejścia-wyjścia są niebuforowane i nie działają na strumienie, lecz na pliki. Funkcjami tymi są open, close, read, write i lseek. Przed wykonaniem jakiejkolwiek operacji na pliku należy go otworzyć. Otwarcia pliku dokonuje funkcja open, w której wywołaniu należy podać nazwę pliku oraz rodzaj otwarcia (tj. odczyt, zapis lub obie czynności jednocześnie). Funkcja open udostępnia liczbę całkowitą stanowiącą identyfikator (deskryptor) pliku. Wywołania read i write działają podobnie do fread i fwrite, przy czym zamiast strumienia podaje się dyskryptor pliku. Operują one bajtami, a więc argument, tj. adres podany w wywołaniu musi być typu *char. Funkcje lseek i close są niebuforowanymi odpowiednikami buforowych funkcji fseek i fclose.

Istnieje kilka przydatnych funkcji działających na teksty. Teksty muszą kończyć się znakiem końca tekstu (NULL czyli '/0'). Funkcja strcat dopisuje do tekstu podanego jako pierwszy kopię tekstu podanego jako drugi. Funkcja strcmp porównuje dwa teksty i udostępnia liczbę całkowitą większą, mniejszą lub równą zeru, zależnie od tego, czy pierwszy z tekstów jest leksykograficznie większy, mniejszy czy równy drugiemu. Funkcja strcpy jest przypisaniem tekstowym, tj. kopię drugiego tekstu umieszcza w miejscu pierwszego. Funkcja strlen oblicza długość tekstu. Funkcja index (rindex) oblicza wskaźnik do pierwszego (ostatniego) wystąpienia podanego znaku w tekście. Funkcja strncat, strncmp i strncpy są odpowiednikami strcat, strcmp i strcpy działającymi na pierwsze n znaków tekstu, gdzie n jest argumentem całkowitoliczbowym tych funkcji. Dwie pierwsze funkcje działają najwyżej na n znaków (gdy teksty są krótsze), natomiast strncpy dokładnie na n znaków, dopełniając teksty krótsze znakami NULL.

KOMPILATOR

Kompilator języka C ma w systemie UNIX nazwę cc. Proste programy składające się z jednego pliku kompiluje się wydając polecenie

```
cc fcpy.c
```

wskutek działania tego polecenia powstaje program wykonywalny o nazwie a.out, o ile plik fcpy.c zawiera program poprawny w C. Przyjęto umownie, że nazwa pliku zawierającego program źródłowy (lub jego część) musi mieć końcówkę .c. Dotyczy to tylko plików wejściowych kompilatora cc, co zresztą jest sprawdzane, natomiast pliki włączane poleceniem # include przez preprocesor mogą mieć dowolne nazwy.

Jeśli program składa się z kilku plików, to wywołanie kompilatora powinno zawierać ich pełną listę, np.

```
cc kalkulator.c stos.c
```

W tym przypadku kompilator tworzy, oprócz pliku a.out, pliki pośrednie — kalkulator.o i stos.o — zawierające tzw. przemieszczalne moduły wynikowe. Moduły te powstają niezależnie, tzn. jeśli w czasie kompilacji wykryto błędy w którymś z plików, to moduł wynikowy nie zostanie utworzony tylko dla tego pliku. Gdy wszystkie moduły zostaną utworzone, następuje faza łączenia (ang. link), w wyniku której powstanie program wykonywalny a.out. W tej fazie mogą być wykryte błędy dotyczące nierozwiązanych odwołań do nazw zewnętrznych. Takie nierozwiązane odwołania powstają wskutek wywołania funkcji w którymś z plików, lub użycia zmiennej zadeklarowanej jako extern, które nie zostały zdefiniowane w żadnym z łączonych plików.

Błędem jest brak w programie funkcji o nazwie main. Konieczność wystąpienia tej nazwy nie wynika z definicji języka, lecz z wymogów polecenia cc. Otóż cc wywołuje program łączący (ang. linker), będący ogólnym programem systemowym. Linker tworząc moduł wykonywalny musi zaznaczyć w nim punkt startowy. Najczęściej programy takie przyjmują domyślnie za punkt startowy pierwszą wykonywaną instrukcję pierwszego z łączonych modułów. Polecenie cc, wywołując linker, wyszukuje wśród plików pośrednich funkcję o nazwie main, którą według przyjętej konwencji uznaje za główną funkcję programu. Punktem startowym programu jest więc pierwsza wykonywana instrukcja tej funkcji. Niemniej program bez funkcji main jest składniowo poprawny i można dlań uzyskać program wykonywalny, ale w inny nieco sposób.

Tworzenie plików pośrednich jest wygodne dla programisty, bowiem w przypadku błędów nie trzeba powtórnie kompilować plików poprawnych. Jeśli np. w kompilacji wystąpiły błędy w pliku stos.c, to plik ten należy poprawić i powtórnie uruchomić kompilator, ale wywołaniem

```
cc kalkulator.c stos.c
```

Polecenie cc ze wszystkich podanych plików wybiera te z końcówką .c i tylko dla nich wywołuje kompilator. Nowo utworzone pliki z końcówką .o oraz pozostałe na liście, które również muszą mieć w nazwie końcówkę o., stanowią wejście dla programu łączącego.

Kompilatory języka C w innych systemach operacyjnych działają również według opisanych zasad.

Polecenie cc może być wywołane z szeregiem tzw. opcji, które zmieniają jego działanie. Przykładowo opcja —lm powoduje włączenie do programu wynikowego funkcji z biblioteki libm. Aby program wykonywalny nie nazywał się a.out, zamazuje on bowiem poprzedni program, należy wydać polecenie z opcją —o, np.

```
cc —o kalkulator stos.o kalkulator.c
```

Program wynikowy otrzyma nazwę kalkulator. Jedną z często używanych jest opcja —c powodująca przerwanie działania polecenia cc po kompilacji, tzn. bez wykonania programu łączącego. Opcji tej wygodnie jest używać do sprawdzania poprawności poszczególnych plików wchodzących w skład dużego programu. Po otrzymaniu wszystkich plików pośrednich można podać polecenie cc dla plików z końcówką .o, co spowoduje wywołanie jedynie programu łączącego, lub — bezpośrednio wywołać ten program poleceniem ld. To drugie polecenie spowoduje utworzenie programu

wynikowego z pominięciem konwencji dotyczącej nazwy main.

W systemie UNIX, oprócz polecenia cc z opcjami, istnieje wiele innych środków programowych związanych z językiem C. W innych systemach, w których zaimplementowano C, wiele z tych środków jest również dostępnych. Najważniejszym z nich jest program lint. Większość kompilatorów C ma słabą diagnostykę błędów. Uznano bowiem, że kompilatory powinny być możliwie małe, aby mogły działać szybko i skutecznie. Diagnostykę natomiast, jako czasochłonny i „pamięciochłonny" element kompilacji, wydzielono w programie lint. Dokonuje on obszernej analizy programu źródłowego, drukując komunikaty nie tylko o błędach, ale również o poprawnych konstrukcjach, które wydają się być pomyłkami. Przykładowo, zamiast

```
if (x & 1 == 0)
```

powinno być

```
if (( & 1) == 0)
```

bowiem pierwsza konstrukcja nie ma sensu. Ponadto lint dokonuje dodatkowej analizy semantycznej, np. sprawdza zgodność typów parametrów wywołania funkcji, oraz — pewnej analizy w celach optymalizacyjnych.

PRZYKŁAD

W charakterze przykładu przedstawimy program kalkulatora obliczający wyrażenia zapisane w odwrotnej notacji beznawiasowej (tzw. polskiej), w której np. wyrażenie

$(12 + 4)/(5 - 2) =$

zapisuje się jako

```
12 4 + 5 2 - / =
```

Kalkulator taki realizuje się przy użyciu stosu, co jest naturalne dla notacji beznawiasowej.

```
#define DLSTOS 100

int sp = 0;                 /*wskaznik stosu*/
double st[DLSTOS];          /*stos*/

double push(w)              /*wrzuc w na stos*/
double w;
{
  if(sp < DLSTOS)
       return(st[sp++] = w);
  else { printf("stos zapelniony\n");
         clear();
         return(0);
       }
}

double pop()                /*zdejmij z wierzcholka sto
  if(sp > 0)
       return(st[--sp]);
  else { printf("stos pusty\n");
         clear();
         return(0);
       }
}

clear()                     /*oproznij stos*/
{
   sp = 0;
}
```

Rys. 1. Plik stos.c

Funkcje push, pop i clear zawarte w pliku stos.c (rys. 1) opisują działanie stosu. Funkcja główna programu znajduje się w pliku kalkulator.c (rys. 3), natomiast funkcję o nazwie getop, której zadaniem jest wczytywanie i rozpoznawanie znaków ze zbioru wejściowego kalkulatora, umieszczono w pliku input.c (rys. 4). Plik kalkulator.def (rys. 2) zawiera definicje stałych. Wydzielenie definicji jest dobrą praktyką, bowiem ewentualne zmiany nie wymagają żadnych zmian w plikach korzystających z tych definicji, a jedynie powtórnej kompilacji.

```
#include <stdio.h>

#define MAXOP 20
#define LICZBA '0'
#define NADMIAR '9'
#define NDFORMAT "%.20s - jest za dlugie\n"
```

Rys. 2. Plik kalkulator.def

Stos zdefiniowano jako wektor elementów typu double. Długość wektora zadano stałą DLSTOS. Wskaźnik stosu sp wskazuje zawsze jego szczyt, tj. pierwsze wolne miejsce. Początkowo ma on wartość 0. Funkcja push wpisuje na szczyt stosu wartość argumentu wywołania, po czym zwiększa wskaźnik stosu o 1, sprawdzając czy nie przekroczył on zakresu wektora. W takim wypadku oczyszcza się stos i wyprowadza tekst „stos zapełniony". Wartością funkcji jest wartość argumentu lub zero (przy pełnym stosie). Funkcja pop działa odwrotnie, tj. obniża stos, a jej wartością jest zawartość szczytu stosu, chyba że stos był pusty (tj. sp == 0). Funkcja clear wymazuje zawartość stosu.

```
#include "kalkulator.def"

main()              /*kalkulator w notacji odwrotnej*/
{
  int rodzaj;
  char oper[MAXOP];
  double op2;

  while ((rodzaj=getop(oper,MAXOP))!=EOF)
    switch(rodzaj)
    {case LICZBA:
          push(atof(oper));
          break;
     case '+':
          push(pop() + pop());
          break;
     case '-':
          op2 = pop();
          push(pop() - op2);
          break;
     case '*':
          push(pop() * pop());break;
     case '/':
          op2 = pop();
          if(op2 != 0.0)
               push(pop() / op2);
          else
               printf("dzielenie przez zero\n");
          break;
     case '=':
          printf("\t%f\n",push(pop()));
          break;
     case 'c':
          clear();
          break;
     case NADMIAR:
          printf(NDFORMAT,oper);
          break;
     default:
          printf("nieznany rozkaz: %c\n",typ);
          break;
     }
}
```

Rys. 3. Plik kalkulator.c

Treścią funkcji głównej jest pętla while wykonywana tak długo, aż wartością funkcji getop będzie stała EOF (koniec pliku). Tak więc, program wejściowy kalkulatora musi być plikiem zakończonym stałą EOF. Po każdorazowym wywołaniu getop dokonywany jest wybór operacji zależnie od wartości funkcji. Kalkulator jest prosty, gdyż operacje są wykonywane dla czterech działań podstawowych: +, —, * oraz /. Pozostałe operacje są następujące: wypisanie wyniku (jest nim zawartość szczytu stosu), opróżnienie stosu, umieszczenie wczytanej liczby na stosie oraz wydruki błędów — gdy pojedynczy argument jest zbyt długi (zapisany przez zbyt wiele znaków), lub gdy wczytany znak jest nieprzewidziany.

```
#include "kalkulator.def"

getop(s)
char s[];
{
   int i, c;
   while((c=getchar()) == ' ' || c == '\t'||c == '\n')
        ;
   if (c != '.' && (c < '0' || c > '9')) /*nie liczba*/
      return(c);
   s[0] = c;
   for(i=1; (c = getchar())>= '0' && c<= '9'; i++)
      if (i<MAXOP)
         s[i] = c;
   if(c == '.'){          /*ulamek*/
      if (i<MAXOP)
         s[i] = c;
      for(i++; (c = getchar()) >= '0' && c <= '9'; i++)
         if (i<MAXOP)
            s[i] = c;
   }
   if (i<MAXOP){          /*liczba poprawna*/
      ungetc(c,stdin);
      s[i]= '\0';
      return(LICZBA);
   } else {               /*liczba za duga*/
      while (c != '\n' && c!= EOF)
         c = getchar();
      s[zakres-1] = '\0';
      return(NADMIAR);
   }
}
```

Rys. 4. Plik input.c

Stosunkowo najbardziej złożona jest funkcja wczytująca getop. Pierwsza pętla tej funkcji służy do przepuszczenia ewentualnych odstępów, znaków tabulacji i zmian wiersza. Jeśli pierwszy wczytany istotny znak nie jest cyfrą ani kropką (jest więc operatorem), to wartością funkcji getop jest jego wartość liczbowa w kodzie ASCII. W przeciwnym wypadku jest to pierwszy znak liczby, tak więc powinno nastąpić wczytanie kolejnych cyfr tej liczby i u-

mieszczenie ich jako znaków w tablicy — parametrze funkcji. Na zakończenie sprawdza się czy liczba nie miała więcej znaków niż dopuszczono i w miejsce ostatniego znaku w tablicy (nie jest on cyfrą) wpisuje się znak NULL. Jeżeli następny znak z wejścia został już wczytany, to trzeba go „cofnąć". Służy temu wywołanie funkcji ungetc, opisanej w pierwszym punkcie. Wartością funkcji staje się stała LICZBA. Jeśli natomiast wczytana liczba jest zbyt długa, to przepuszczone zostaną kolejne znaki z wejścia, aż do znaków nowego wiersza lub końca pliku. Wartością funkcji jest wtedy stała NADMIAR. Oprócz swej wartości funkcja getop przekazuje funkcji wywołującej jeszcze jeden obiekt. Parametrem getop jest tablica (adres), a więc operacje wykonywane na tablicy st są w rzeczywistości wykonywane na tablicy oper funkcji main. Zawartość tej tablicy (jest nią tekst reprezentujący kolejne znaki liczby) zostaje przetworzona przez funkcję biblioteczną atof na reprezentację zmiennoprzecinkową tej liczby.

Można zastanowić się nad udoskonaleniem i rozbudowaniem kalkulatora. W funkcji main niepokój budzi działanie wykonywane dla operatora =. Wywołanie push (pop ()) można zastąpić wywołaniem nowej funkcji top (), której definicję

```
top ( )
{
return(st[sp — 1]);
}
```

należy umieścić w pliku stos.c. Bezpośrednie użycie zmiennych st i sp w funkcji main jest oczywiście możliwe, o ile zadeklaruje się je jako zewnętrzne. Niemniej celowe jest, aby kalkulator nie był zależny od konkretnej realizacji stosu.

Rozbudowa kalkulatora o nowe operatory polega na dodaniu nowych ewentualności case z nowymi operacjami. Jeżeli kalkulator ma szereg miejsc pamięci do przechowywania wyników częściowych, to dla prostoty można je zapisywać w programie wejściowym kalkulatora jako pojedyncze małe litery. Do operowania nimi służą dwie operacje: '<' oznaczająca „zapamiętaj", i '>' — „przypomnij". Są to operacje jednoargumentowe o argumencie — symbol miejsca pamięci. Aby zrealizować to rozszerzenie można dodać deklaracje

```
double memory [26];
int ind;
```

oraz nowy fragment instrukcji switch

```
case 'a':case 'b': ...case 'z':
    ind = c — 'a'; break;
case '>': push(memory[ind]); break;
case '<': memory[ind] = top(); break;
```

Można również uprościć wczytywanie liczb. Zamiast wczytywać po jednym znaku funkcją getchar, a następnie dokonywać konwersji wczytanego tekstu funkcji atof, można użyć funkcji scanf, która wczyta wszystkie znaki liczby i dokona konwersji (rys. 5).

```
#include "kalkulator.def"

getop(f1)
double *f1;
{
    char s[256], cf
    while((c=getchar()) == ' ' ||c == '\t'||c == '\n')
        ;
    if (c != '.' && (c < '0' || c > '9')) /*nie liczba*/
        return(c);
    ungetc(c,stdin);
    scanf("%20lf",f1);
    c=getchar();
    if (c != '.' && (c < '0' || c > '9'))      /*liczba poprawna*/
    {  ungetc(c,stdin);
       return(LICZBA);
    } else {              /*liczba za duga*/
       gets(s);
       return(NADMIAR);
    }
}
```

Rys. 5. Nowy plik input.c

Nowa funkcja getop ma parametr typu wskaźnikowego, a więc jej wywołanie należy zmienić na getop (& wynik), gdzie wynik jest zmienną typu double, której definicja powinna zastąpić definicję oper. Działanie dla operatora '=' też się uprości do postaci push (wynik).

amepol

AMEPOL — Przedsiębiorstwo Zagraniczne w Polsce oferuje do sprzedaży po nowych atrakcyjnych cenach niezawodne

PÓŁPRZEWODNIKOWE PAMIĘCI OPERACYJNE do następujących komputerów i minikomputerów:

- **ODRA 1305 o pojemności 64K, 128K, 192K, 256K, 512K,**
- **ODRA 1325 o pojemności 64K i 128K,**
- **RIAD R-32 o pojemności 256KB, 512KB, 768KB, 1MB i 2MB,**
- **MERA 400 o pojemności 64K, 128K, 256K, 512K,**
- **MERA 9150 o pojemności 64K,**
- **NOVA 1200 o pojemności 64K,**
- **PDP 11 o pojemności 64K i 128K,**
- **SM 4 o pojemności 64K i 128K.**

Pamięci półprzewodnikowe P.Z. AMEPOL skonstruowane są w oparciu o elementy DRAM VLSI produkcji zachodniej, co zapewnia dużą szybkość działania i znikomy pobór mocy z jednego źródła zasilania. Pamięci mogą współpracować z pamięciami ferrytowymi lub pracować samodzielnie.

Do końca bieżącego roku wdrożone będą do produkcji kontrolery dyskowe i procesory komunikacyjne. Przewidujemy również produkcję innych urządzeń informatyki.

Oferujemy krótkie terminy dostaw i instalacji. Każdy wyrób posiada roczną gwarancję i pełny serwis pogwarancyjny.

P.Z. AMEPOL uprzejmie prosi wszystkich zainteresowanych naszymi wyrobami o kontakty listowne, telefoniczne lub teleksowe.

Nasz adres:
Plac Żelaznej Bramy 1
00-136 Warszawa

PRZEDSIĘBIORSTWO ZAGRANICZNE

amepol

Zakład Elektroniki i Aparatury Medycznej

Telefon: 20-34-75, 20-45-05
Teleks: 812539 apol pl

EO/377/K/85

Napór czytelników-właścicieli ZX SPECTRUM na mikroKLAN nie tylko nie słabnie, ale wręcz nasila się. Dlatego też poświęcamy tyle miejsca temu mikrokomputerowi w tym numerze. O ile wielbicieli sprzętu chyba w pełni usatysfakcjonuje publikowany schemat ZX SPECTRUM, to dla zadowolenia tych co „grzebią się" w oprogramowaniu przygotować musieliśmy coś równie atrakcyjnego. Nie możemy wprawdzie opublikować „zdisasemblowanego" ROM-u (zajmuje to wcale nie małą książkę), ale za to publikujemy pierwszą część materiału wyjaśniającego jak uzupełniać to co jest we wspomnianym ROM-ie.

Możliwość rozszerzenia podstawoweko zestawu instrukcji może okazać się wyjątkowo użyteczna przy wszelkich niestandardowych zastosowaniach. Dotyczy to zarówno przystawek sprzętowych, jak i pewnych zastosowań programowych, w których BASIC nie wystarcza, lecz z różnych powodów (miejmy nadzieję, że nie tylko z lenistwa) nie chcemy sięgać po kompilatory innych języków.

Jak rozszerzyć BASIC ZX SPECTRUM

Wśród wad, które zarzuca się ZX SPECTRUM, dosyć często wymieniany jest ograniczony zestaw instrukcji. Oczywiście niedociągnięcia nie czekały zbyt długo na poprawienie. Powstały odpowiednie programy — na przykład — BETA BASIC z jego ostatnią wersją V 1.8. czy SUPERCODE 2. Naprzeciw oczekiwaniom klientów wyszła również sama firma Sinclair, umożliwiając przy wykorzystaniu INTERFACE 1 tworzenie nowych instrukcji, definiowanych przez samego użytkownika.

Okazuje się, że jest to możliwe również bez tej kosztownej przystawki — i to dla obu wersji komputera: 16 K i 48 K. Koncepcja rozszerzania zestawu instrukcji jest stosunkowo prosta, choć jej praktyczna realizacja niestety do najprostszych nie należy.

Tym, którzy po lekturze tego artykułu zapragnęliby dodać kilka funkcji użytkowych do swego komputerka, radziłbym wcześniej zapoznać się z możliwościami wspomnianego wcześniej programu BETA BASIC. Z pewnością łatwiej będzie zdobyć ten program, niż „na piechotę" wprowadzać duże ilości kodu maszynowego. Opisana metoda może natomiast przydać się dla uzupełnienia zestawu instrukcji w języku BASIC ZX SPECTRUM o nietypowe funkcje (np. graficzne). Być może niektórzy już domyślają się, jak można rozszerzyć zestaw instrukcji ZX SPECTRUM:

— po pierwsze, należy spowodować wygenerowanie przez system błędu składniowego (ang. syntax error),

— po drugie, „przechwycić" obsługę tego błędu powodując, że zamiast wyprowadzenia komunikatu o błędzie — wykonana zostanie sekwencja instrukcji dająca odmienny, z góry założony efekt.

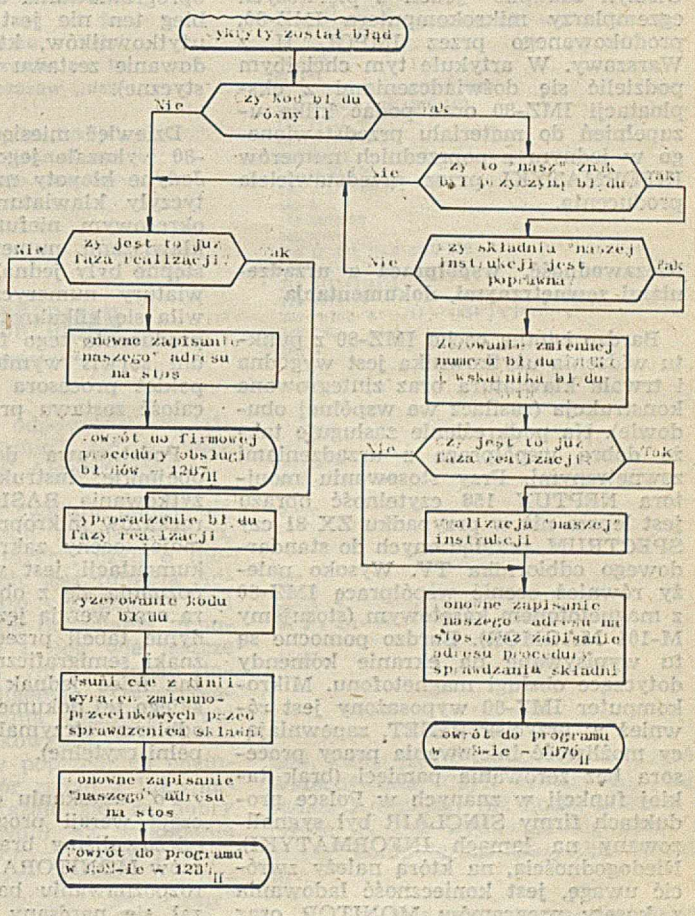
Błąd składniowy można spowodować dwojako (użytkownik ma pełną swobodę wyboru):

A) użycie dowolnego ze słów kluczowych w świadomie błędny sposób (np. DRAW $); nie ma to oczywiście wpływu na normalne użycie i działanie „zniekształcanych" instrukcji w innych miejscach programu.

B) użycie dowolnego ze znaków otrzymywanych przez kombinację z klawiszem SHIFT (np. *, $) i wprowadzenie nazwy nowej instrukcji, litera po literze, np. *OWAL, *ELIPSA. Znaki uzyskiwane przy użyciu klawisza SHIFT służą do wyjścia poza tryb K (tryb wprowadzania instrukcji) po to, aby możliwe było podawanie pojedynczych liter (w trybie K zamiast litery pojawi się od razu całe słowo kluczowe).

Wykrycie nieprawidłowości powoduje przejście do procedury obsługi błędu. Adres tej procedury zapisany jest w lokacji pamięci wskazywanej przez zmienną systemową ERR-SP (adres: 23613/14). Stwierdzenie błędu, zarówno w fazie sprawdzania poprawności składni wprowadzanej instrukcji, jak i w fazie realizacji, powoduje następujące akcje wykonywane przez program zapisany w pamięci ROM:

- przepisanie zawartości zmiennej CH-ADD (adres: 23645) do wskaznika błędu XPTR (adres: 23647); wskaźnik zawiera adres określonej przez interpreter błędnej pozycji w analizowanej linii,
- do zmiennej ERRNR (adres: 23610) zapisywany jest kod błędu wg następującej zasady: numer błędu jest o jeden mniejszy niż numer raportu błędu, który pojawia się po wystąpieniu błędu w fazie realizacji programu; dla numeru błędu 255 występuje raport 0 — O.K., dla błędu numer 11 raport C — nonsense in BASIC; taki właśnie raport pojawiać się będzie przy wykryciu nierozpoznanej przez interpreter instrukcji i to zarówno w fazie sprawdzania poprawności składni jak i w fazie wykonania,
- wskaźnik stosu ładowany jest zawartością zmiennej ERR SP,
- zerowane są zarówno pamięć jak i stos kalkulatora zmiennoprzecinkowego,
- wykonanie instrukcji RET powoduje skok pod adres przechowywany na stosie. Podczas sprawdzania poprawności składni adres ten równy jest 12B7H, zaś znajdujący się tam



Schemat blokowy opisywanego algorytmu

program powoduje wyświetlenie całej linii ze znakiem zapytania na błędnej pozycji. W fazie realizacji programu adres ten równy jest 1303H, zaś znajdujący się tam program spowoduje przerwanie wykonywania programu użytkowego i wyświetlenie odpowiedniego komunikatu błędu.

Umieszczenie na stosie adresu „własnego" programu (powinien być on zapisany powyżej adresu RAMTOP + 1) skieruje działanie systemu operacyjnego do tego właśnie programu, tak jakby wykonywana była standardowa obsługa błędu.

Na rysunku przedstawiony został schemat blokowy algorytmu rozszerzania zestawu instrukcji języka BASIC dla ZX SPECTRUM. W następnej części artykułu zamieszczone zostaną wydruki programów tworzących nowe instrukcje.

MAREK GÓRECKI
Warszawa

Mikroinformatyczny boom zgotowany nam przez firmy polonijne i rzemieślnicze nadal trwa. Zamiast stopniowego zaspokajania głodu możemy jednak obserwować ciągły wzrost apetytów. W rezultacie mikrokomputery są dosłownie rozchwytywane i to na zasadzie: brać to co jest, potem będziemy zastanawiać się co z tym zrobić.

Chociaż zapewne wszyscy zdają sobie sprawę jak zgubne rezultaty może to przynieść, to w obliczu możliwości kupienia za złotówki mikrokomputera tracą głowę. Publikując na łamach mikroKLANU opisy różnych krajowych konstrukcji, liczymy więc bardziej na oddziaływanie podświadomości, która skłoni (lub powstrzyma) do zakupu określonego mikrokomputera do określonych celów. Sądzimy, że oprócz ogólnych opisów dostarczanych nam przez producentów bardzo przydatne będą opinie użytkowników, czyli tych, którzy przymierzyli daną konstrukcję do określonych zadań i przekonali się czy to jest właśnie „ten" komputer jakiego potrzebują.

Zapraszamy do udziału w dyskusji.

Doświadczenia z eksploatacji IMZ-80

W czerwcu ubiegłego roku ZETO Olsztyn zakupiło jeden z pierwszych egzemplarzy mikrokomputera IMZ-80, produkowanego przez IMPOL II z Warszawy. W artykule tym chciałbym podzielić się doświadczeniami z eksploatacji IMZ-80 oraz podać kilka uzupełnień do materiału przedstawionego w jednym z poprzednich numerów INFORMATYKI przez przedstawiciela producenta.

Niezawodność, współpraca z urządzeniami zewnętrznymi, dokumentacja

Bardzo istotną cechą IMZ-80 z punktu widzenia użytkownika jest wygodna i trwała klawiatura oraz zintegrowana konstrukcja (zasilacz we wspólnej obudowie). Na podkreślenie zasługuje także dobra współpraca z urządzeniami zewnętrznymi. Przy stosowaniu monitora NEPTUN 156 czytelność obrazu jest lepsza niż w przypadku ZX 81 czy SPECTRUM, przyłączonych do standardowego odbiornika TV. Wysoko należy również ocenić współpracę IMZ-80 z magnetofonem kasetowym (stosujemy M-101 MAGMOR). Bardzo pomocne są tu wypisywane na ekranie komendy dotyczące obsługi magnetofonu. Mikrokomputer IMZ-80 wyposażony jest również w przycisk RESET, zapewniający możliwość inicjowania pracy procesora bez zerowania pamięci (brak takiej funkcji w znanych w Polsce produktach firmy SINCLAIR był sygnalizowany na łamach INFORMATYKI). Niedogodnością, na którą należy zwrócić uwagę, jest konieczność ładowania z kasety programów MONITOR oraz BASIC. Jednak producent zapowiedział już możliwość wpisania podstawowego oprogramowania do pamięci stałej (zabieg ten nie jest potrzebny dla tych użytkowników, którzy planują rozbudowanie zestawu IMZ-80 o dyski elastyczne).

Dziewięć miesięcy eksploatacji IMZ-80 wykazało jego dużą niezawodność. Jedyne kłopoty natury technicznej dotyczyły klawiatury. Polegały one na okresowym niefunkcjonowaniu linii z klawiszami numerycznymi (cyfry dostępne były jednak nadal na polu klawiatury numerycznej). Usterka pojawiła się kilkakrotnie. Po zgłoszeniu do producenta tego faktu w ciągu dwóch dni serwis wymienił klawiaturę oraz pakiet procesora i od tego momentu całość zestawu pracuje bez zarzutu.

Podstawowa dokumentacja IMZ-80 obejmuje: instrukcję obsługi, opis użytkowania BASIC IMZ-80 oraz listę rozkazów mikroprocesora Z-80. Według mojej oceny zakres merytoryczny dokumentacji jest wystarczający do zapoznania się z obsługą mikrokomputera oraz wersją języka BASIC (brak jedynie tabeli przedstawiającej dostępne znaki semigraficzne). Zastrzeżenia można mieć jednak do poziomu edytorskiego tej dokumentacji (może następni odbiorcy otrzymali egzemplarze już w pełni czytelne).

Po uzyskaniu od producenta „roboczej" wersji programu ASSEMBLER odczuwaliśmy brak opisów podprogramów MONITORA. Przy ich dokładnym rozpoznawaniu bardzo przydatny okazał się napisany przez nas wcześniej (w języku BASIC) DISASSEMBLER. Pomocą służył również konstruktor IMZ-80. Pierwsze miesiące eksploatacji były dla firmy IMPOL II i ZETO Olsztyn okresem zbierania i wymiany doświadczeń. M.in. na podstawie naszych sugestii producent wprowadził kilka usprawnień do oprogramowania IMZ-80. Ostatnio wraz z „handlową" wersją programu ASSEMBLER IMZ-80 otrzymaliśmy pełną dokumentację obejmującą opisy programów ASSEDIT, COMPILER i DEBUGGER oraz istotnych podprogramów zawartych w programie MONITOR. Załącznikiem jest również lista rozkazów mikroprocesora Z-80 w porządku alfabetycznym i szesnastkowym (w takim formacie działa DEBUGGER).

Mylące jest użycie przez przedstawiciela producenta w swoim artykule nazwy MACROASSEMBLER, w odniesieniu do zestawu trzech programów zapewniających możliwość tworzenia i testowania oprogramowania w języku ASSEMBLER. Wprawdzie możliwe jest stosowanie pewnej grupy pseudorozkazów (m.in. zapewniających wygodną organizację pamięci), ale w sumie nazwa MACROASSEMBLER jest użyta na wyrost. Tworzenie programów na tym poziomie byłoby szybsze, gdyby przynajmniej funkcje ASSEDIT i COMPILER zawarte były w jednym programie.

BASIC IMZ-80

Dla czytelników mikroKLANU będzie chyba dogodne porównanie istotnych elementów języka BASIC IMZ-80 z implementacją tego języka dla ZX SPECTRUM.

W IMZ-80 wygodniejszy jest proces tworzenia, a przede wszystkim poprawiania programów. Do tego celu służy edytor ekranowy umożliwiający dokonywanie zmian w programie poprzez operacje na tekście wyświetlonym na ekranie. Po przekroczeniu 40 znaków w wierszu, automatycznie dołączony zostaje następny wiersz zapewniając możliwość tworzenia programów w logicznych liniach zawierających maksymalne 79 znaków. W procesie tworzenia i testowania programów można korzystać z kilku dyrektyw, które nie występują w ZX SPECTRUM: AUTO (automatyczne numerowanie wierszy z zadanym krokiem), DELETE n-m (kasowanie fragmentu programu), TRON (śledzenie przebiegu programu — wypisywanie na ekran numerów kolejno wykonywanych linii).

Poniżej publikujemy informację o wydarzeniu, które może okazać się przełomowe dla rozwoju mikroinformatyki lub też... zaginąć w pomroce dziejów. Czas pokaże. Na razie możemy zauważyć (z pozycji kibica — mikroKLAN nie jest przecież klubem), że publikowana deklaracja ma wiele cech wspólnych z tym co pisaliśmy wprowadzając wkładkę mikroKLAN. Nie to jest jednak źródłem aprobaty jaką darzymy nową inicjatywę. Jesteśmy przekonani, że jedynym sposobem na uchronienie mikroinformatyki od możliwych dewiacji (jakim uległo wiele innych dziedzin naszego życia) jest ruch masowy. Kluby i zrzeszająca je Federacja wydają się na dziś jedyną możliwą formą takiego ruchu.

Polska Federacja Klubów Mikroinformatycznych

„Mikroinformatyka" to pojęcie nie zdefiniowane jeszcze w żadnym słowniku języka polskiego. A jednak przenika ono coraz trwalej w sferę naszej świadomości jako nowe, odrębne od dotychczasowych zjawisko. Fenomen mikroinformatyki bez wątpienia ma swoje źródła w mikroelektronice i tradycyjnej informatyce, wymaga jednak również uwzględnienia, że jest to zarazem masowo upowszechniana metoda konstruktywnego rozumowania.

Chociaż mikroinformatyka w Polsce to ciągle jeszcze w dużej mierze nowa moda, lecz z rodzaju tych, na których może bardzo wiele zyskać całe społeczeństwo.

Zaleganie na tyłach światowego postępu niesie ze sobą również pewną szansę, jak dotąd nie dość docenianą, na zmarnowanie której w obecnej sytuacji po prostu nas nie stać. Śledząc drogę światowego rozwoju mikrokomputeryzacji powinniśmy zwrócić uwagę na wszelkie popełnione dotąd błędy tak aby wybrać najlepszą z tysiąca możliwych ścieżek. Konstruktywnym przejawem troski o wybór pomyślnej drogi rozwoju mikroinformatyki w naszym kraju jest zamysł utworzenia POLSKIEJ FEDERACJI KLUBÓW MIKROINFORMATYCZNYCH.

Federacja jako porozumienie i dobrowolne zrzeszenie wszelkich powstających na terytorium Polski klubów mikrokomputerowych byłaby w stanie przełamać barierę informacyjną (poprzez dysponowanie własnym bankiem informacji) co ma kluczowe znaczenie dla zapewnienia prawidłowego rozwoju mikroinformatyki.

W założeniach, Federacja jako reprezentantka interesów klubów (czyli grup społecznych) i inicjatorka ich powstawania realizowałaby postawione sobie cele poprzez organizowanie spotkań, pokazów, seminariów i kursów oraz wszelkie inne dostępne środki przekazywania informacji.

29 czerwca 1985 roku wybrano tymczasowe władze Federacji (nie jest ona jeszcze oficjalnie zarejestrowana). Wybory generalne odbędą się około 29 listopada 1985 w czasie trwania „Dni Klubów Mikroinformatycznych" w ramach MIKROEXPO '85.

dokończenie na str. 18

Podstawowy kurs Akademii mikroKLANU zmierza ku końcowi. Czy dołożyliśmy własną cegiełkę do rozwijania umiejętności programowania — czas pokaże. Został nam jeszcze jeden odcinek a potem... zobaczymy. Wszystko zależy od reakcji Czytelników. Były już wprawdzie postulaty kursu... języka C, ale jest to chyba propozycja przedwczesna — zbyt mało mikrokomputerów w naszym kraju „rozumie" ten język. Wśród innych propozycji wymieniany był też język LOGO — bez wątpienia bardzo ciekawy. Sądzimy jednak, że tu z kolei sensowna byłaby znacznie szersza popularyzacja (szkoły!), niż pozwala na to skromny nakład INFORMATYKI. Odłożymy więc chyba LOGO do czasów „usamodzielnionego" mikroKLANU — kiedyś to w końcu musi nastąpić.

Nie zamierzamy oczywiście zerwać z publikacjami dotyczącymi języka BASIC. Nadal oczekujemy na ciekawe procedury i podprogramy. Mamy jednak nadzieję, że będą one pisane w eleganckim stylu, tak jak to zalecano „na wykładach" Akademii.

Akademia mikroKLANU (7) ACKERMAN

```
LIST

5 REM OBLICZANIE FUNKCJI ACKERMANA METODA LISTY CELOW.
10 DIM J(1000)
20 INPUT "J=";J(1)
30 IF J(1)<0 OR INT(J(1))<>J(1) THEN GOTO 20
40 INPUT "K=";K
50 IF K<0 OR INT(K)<>K THEN GOTO 40
60 J1=J(1)
70 KS=K
80 I=1
90 IF J(I)=0 THEN GOTO 180
100 IF K=0 THEN GOTO 150
110 I=I+1
120 J(I)=J(I-1)
130 K=K-1
140 GOTO 100
150 J(I)=J(I)-1
160 K=1
170 GOTO 90
180 K=K+1
190 IF I=1 THEN GOTO 230
200 I=I-1
210 J(I)=J(I)-1
220 GOTO 90
230 PRINT "A(";J1;",";KS;")=";K
240 END
READY.
```

```
RUN
J=? 2
K=? 23
A( 2 , 23 )= 49

READY.
```

```
RUN
J=? 3
K=? 2
A( 3 , 2 )= 29

READY.
```

Wprawdzie w języku BASIC nie mamy możliwości rekurencyjnego definiowania funkcji, ale możemy tę trudność ominąć używając tzw. listy celów. Tworzymy sekwencję par liczb j_i, k gdzie każda para jest kolejnym celem obliczeń. Gdy $i = 1$ oraz $j_i = 0$, wtedy policzyliśmy już wszystkie potrzebne cele i znamy odowiedź końcową czyli k. Oczywiście konkretna postać listy celów wynika z algorytmu Ackermana. Najlepiej ilustruje to program szkolny (wydruk).

Jeżeli puścimy nasz program, to już dla niewielkich liczb ($j = 4$, $k = 2$) otrzymamy błąd przekroczenia wartości wskaźnika wektora J. Zwiększamy więc wymiar wektora J, aż do granic pamięci naszego komputera (np. DIM J(100000)). Rezultat — ten sam tyle, że po bardzo, bardzo długim czasie! A przecież dla niektórych liczb otrzymywaliśmy poprawne odpowiedzi, co łatwo sprawdzić „ręcznie". Gdzie więc pies pogrzebany? W Algorytmie!

Nasza lista celów świetnie nadaje się do obliczania funkcji rekurencyjnych, ale takich „normalnych", czyli o niewielkich wartościach. Zaś funkcja Ackermana jest bardzo perfidna — jest to jakby uogólnienie dodawania, mnożenia, potęgowania i Bóg wie jeszcze czego! Łatwo się o tym przekonać stosując własną inteligencję i wyprowadzając tzw. równania różnicowe (nie mylić z różniczkowymi) — patrz wspomniana już w poprzednich odcinkach książka „Zbiór zadań z informatyki", str. 227. Okazało się, że dobry algorytm nie pozwolił na wykonanie obliczeń w skończonym czasie i w ograniczonym obszarze pamięci. Tu nasuwa się uwaga ogólna: „skończony" w dziedzinie programowania, to znaczy taki, który jest mniejszy od wymiarów Wszechświata. Czyli krótszy niż wiek świata i mniejszy niż liczba atomów w tymże. A już wydawało się nam, że dzięki komputerom wszystko potrafimy...

JAKUB TATARKIEWICZ

W mikroKLANIE 9 zamieściliśmy test pozwalający w pewnym zakresie wnioskować o dokładności numerycznej komputera. Dodajmy, że w stosunkowo ograniczonym zakresie, gdyż jak wykazało życie (a raczej czytelnicy w telefonach do redakcji) — wyłożył się na nim choćby minikomputer z serii PDP, dając zaskakująco słabe wyniki. Czy to oznacza, że PDP-y zostały już pokonane przez mikrokomputery? Chyba jeszcze nie całkiem. Każdy test jest w pewnym zakresie subiektywny — coś uwypukla, coś zaniedbuje. Nie należy jednak załamywać rąk: jeśli wykonujemy obliczenia numeryczne, powinniśmy w jakiś sposób szacować rzetelność wyników. Pozostaje więc... doskonalić testy. Tym razem prezentujemy test standardowych funkcji. Czytelników, którzy wymyślili lub znają lepsze testy, zapraszamy do podzielenia się z innymi za pośrednictwem naszych łamów.

Test numeryczny

Publikujemy test, którego głównym celem jest porównanie dokładności obliczeń numerycznych standardowych funkcji. Pomysł zaczerpnęliśmy z listu Martena den Boera do redakcji PHYSICS TODAY (listopad 1984, str. 154), nieco modyfikując zarówno sam test, jak i obliczanie wartości **D**.

A oto lista programu:

```
10 P2 = 1.57
12 DEF FN A(X) = ATN (X / ( SQR ( - X * X + 1) + 1E - 35))
13 DEF FN B(X) = 2 * ATN (1) - FN A(X)
20 INPUT "ILOSC ITERACJI";N
30 D = 0.0
40 A = 0.0
50 S = P2 / N
60 FOR I = 1 TO N
70 A = A + S
80 B = SIN (A)
90 C = FN A(B)
100 B = COS (C)
110 C = FN B(B)
120 B = TAN (C)
130 C = ATN (B)
140 B = EXP (C)
150 C = LOG (B)
160 B = SQR (C)
170 C = B ^ 2
180 D = D + (A - C) * (A - C)
190 NEXT I
200 D = SQR (D / N)
210 PRINT "DOKLADNOSC = ";D
220 END
```

Test zapisany dla APPLE II Europlus w języku MBASIC

Przepis na uruchomienie: Wpisz program do pamięci Twego komputera. Jeżeli BASIC, którego używasz, nie ma funkcji arcsin oraz arccos, to użyj linii 12 oraz 13, a linie 90 i 110 pozostaw bez zmiany. Jeżeli natomiast dostępna Ci wersja języka BASIC ma wspomniane funkcje trygonometryczne, to wstaw je w liniach 90 i 110, a opuść linie 12 oraz 13.

Wpraw w ruch program dla **N = 17**, otrzymany czas (w sekundach) odpowiada przybliżonemu czasowi (w minutach), jaki otrzymasz dla **N = 1000**. Przeprowadź tekst dla **N = 1000** i zanotuj wynik (czas oraz wartość **D**). Porównaj swój komputer z tabelą.

Wnioski potrafi wyciągnąć chyba każdy Czytelnik mikroKLANU: MERITUM nadaje się do gier, a IBM PC z koprocesorem — do obliczeń naukowych. Chyba że posiadamy MERITUM i musimy odwrócić macierz 45×45, czego Czytelnikom mikroKLANU nie życzy

JAKUB TATARKIEWICZ

Autor serdecznie dziękuje wszystkim, którzy udostępnili swoje komputery dla przeprowadzenia powyższego testu.

Wyniki testu dla różnych mikrokomputerów

Komputer	Język	Czas (s)	D	Uwagi
IBM PC	MBASIC	1 7	4.139E-06	sic!
	TURBO PASCAL	10,7	8.632E-16	koprocesor 8087
	turbo pascal	439	2.836E-10	
TI 99/4A	Extended BASIC	1318	8.013E-12	
HP 85	HP BASIC	362	1.149E-11	ma ASN i ACS
HP 75C	HP BASIC	423	1.149E-11	ma ASN i ACS
HP 41C	RPN	3252	1.869E-09	ma ASN i ACS
SINCLAIR ZX81	BASIC	957	8.828E-09	ma ASN i ACS
	BASIC	4020	8.828E-09	fast mode slow mode
SINCLAIR SPECTRUM	BASIC	995	8.818E-09	ma ASN i ACS
	PASCAL	69	2.948E-05	
APPLE II+	MBASIC	732	1.145E-08	
	MBASIC kompilowany	502	1.147E-08	
	PASCAL	837	1.561E-06	
NEPTUNE 184	BASIC	835	1.145E-08	
CBM 64	BASIC	835	1.145E-08	
VIC 20	BASIC	678	1.148E-08	
EPSON HX-20	MBASIC v.1.0	965	1.606E-06	
MERITUM	BASIC	623	1.609E-06	brak DEF FN
TRS 80/I	BASIC (level II)	1020	1.609E-06	
	FORTRAN	400	4.733E-06	nieoptymalizowany
OSBORNE	BASIC	333	4.628E-06	
Twój komputer	.....	.....	.....	.....

dokończenie ze str. 17

Pierwszą zakrojoną na szerszą skalę imprezą organizowaną przez Federację mają być DNI KULTURY MIKROINFORMATYCZNEJ MIKROEXPO '85. Ten swego rodzaju chrzest bojowy ma być sprawdzianem sensu istnienia Federacji a przede wszystkim ludzi i gotowych do współpracy organizacji, czyli wszystkich nieobojętnych na stan mikroinformatyki w Polsce.

Według wstępnych planów przewidziane zostały następujące prezentacje: grafika komputerowa, muzyka komputerowa, krajowa i zagraniczna oferta sprzętowa, dorobek poszczególnych klubów, filmy o aktualnym stanie mikroinformatyki na świecie (EXPO 85).

Planowane jest też zorganizowanie cyklu seminaryjnego, w którym m.in. odbędzie się spotkanie na temat języka LOGO, jako narzędzia do celów edukacyjnych i profesjonalnych. Będą też tematy związane z prezentacjami.

Do udziału w MIKROEXPO '85 zaproszeni będą goście z krajów RWPG, Europy Zachodniej i Stanów Zjednoczonych. W czasie Dni Kultury opracowany zostanie raport o stanie polskiej kultury mikroinformatycznej. Podjęta zostanie w nim próba ukazania roli różnych instytucji w dotychczasowym rozwoju mikroinformatyki.

Pragniemy zaapelować do wszystkich, którzy działają w różnych klubach mikrokomputerowych (lub mających zamiar w ten sposób się zorganizować) o aktywne przyłączenie się do naszych działań. Mamy nadzieję, że dzięki powstaniu Federacji wiele trudności przestanie być problemem, a dostęp do różnych typów mikrokomputerów nie będzie nieosiągalną, niejasną w czasie i przestrzeni mrzonką, lecz realną możliwością dla każdego zainteresowanego.

PIOTR TYMOCHOWICZ
ul. Renesansowa 5 m. 10
01-905 Warszawa, tel. 35-77-60

W Życiu Warszawy (nr 37, 13 luty 1985) przeczytałem, że Zrzeszenie MERA rozpoczyna wyposażanie warszawskich szkół w komputery. Informacja ta w pierwszej chwili mnie ucieszyła, ale radość moja natychmiast zgasła po przeczytaniu następnego akapitu, w którym podano, że specjaliści z ZETO w Łodzi na tygodniowym kursie będą szkolić nauczycieli. Każdy absolwent tego kursu będzie już umiał pisać proste programy i uczyć innych. Doprawdy trudno tutaj nie załamać rąk. Stąd też postanowiłem zaprezentować w dziale „Dydaktyka" poszerzoną wersję poglądów dr. Czyżo, przedstawioną w nr 4/1985 INFORMATYKI.

Trzeba sobie zdawać sprawę, że rozpoczyna się batalia o kształt wiedzy informatycznej kolejnego pokolenia, które za kilka lat wstąpi w progi uczelni. Jeżeli już teraz nie rozpocznie się starań o prawidłowe i rzetelne szkolenie, to w przyszłości całe odium zniechęcenia kolejnego pokolenia do informatyki spadnie na nas. Dlatego też zachęcam do pilnego śledzenia dyskusji o mikrokomputerach w szkole, prowadzonej na łamach tegorocznej POLITYKI, i jeżeli to możliwe uświadomienie wszystkim, że nawet BASICU nie powinien nauczać ten, który tylko tydzień się go uczył. No, ale oddajmy głos autorowi tekstu. (Red.).

Nauczyciel — uczeń — mikrokomputer

Jeśli się przyjrzeć zjawisku mikrokomputeryzacji spokojnie, to rysuje się obraz następujący: ogromna większość entuzjastów mikrokomputerowych to dzieci i młodzież; większość programów eksploatowanych na mikrokomputerach, to gry i zabawy, przeznaczone głównie dla dzieci i młodzieży.

Czy to dobrze, czy źle? Naturalnie, że dobrze, jeśli dziecko umie obchodzić się z mikrokomputerem. Wtedy, na przykład będzie miało większą wprawę w pisaniu na maszynie. A jest to umiejętność o podstawowym znaczeniu, której niestety nie zdobywa się w szkole.

Rozrywka komputerowa oczywiście nie jest w niczym gorsza od innych rozrywek, przynajmniej na razie. A często bywa lepsza, gdy zostanie zaprojektowana z pewnym celem dydaktycznym. Może trzeba to podkreślić wyraźniej — prawie wszystkie rozrywki komputerowe, nawet jeśli są tylko rozrywkami, nie są niczym złym. Dziecko może pobawić się w sposób dość atrakcyjny i bezpieczny. I to już samo w sobie stanowi wartość. Ale oczywiście na tym sprawa się nie kończy, a raczej trzeba przyjąć, że dopiero się zaczyna. Po jakimś czasie zabawy te stają się mniej atrakcyjne i dziecko oczekuje innych rozrywek.

W tej sytuacji pojawia się pierwsza poważna szansa na umiejętne, w sensie pedagogicznym, wykorzystanie tego zainteresowania i zapotrzebowania na nowe rozrywki komputerowe. W pewnym momencie pojawia się też ciekawość i chęć tworzenia własnych rozrywek, a więc programów, bądź ingerowania w programy dotychczas eksploatowane. Jednym słowem, dziecko trzeba nauczyć programowania w języku dostępnym na danym mikrokomputerze. Z pedagogicznego punktu widzenia jest to sytuacja bardzo pożądana, gdyby nie pewne problemy, na które chciałbym zwrócić uwagę.

Pierwszy problem wiąże się z językiem programowania. Znakomita większość rozrywek komputerowych jest opracowana w języku BASIC, a tylko niektóre, opracowane — w językach asemblerowych. Programowanie w języku asemblera jest bardziej kształcące, gdyż pozwala lepiej zrozumieć działanie komputera, ale z drugiej strony, jest zajęciem bardzo obciążającym psychicznie i mam poważne obawy, czy powinno się obciążać dziecko takim wysiłkiem.

Wbrew pozorom, dużo gorszą sytuację mamy w przypadku programowania w języku BASIC. Jeden z najwybitniejszych informatyków, Edsger Dijkstra, nazwał w swoim czasie BASIC językiem antydydaktycznym. Jak powszechnie wiadomo w środowisku informatycznym, a powinni o tym wiedzieć także pedagodzy, pierwszy poznany i używany język programowania ma znaczny wpływ na podejście człowieka do rozwiązywania problemów. Inaczej mówiąc, to czego nauczymy się przy pierwszym języku programowania, tak utrwala się w umyśle, iż trudno się od niego w przyszłości uwolnić. Gdybyśmy zatem chcieli uczyć dziecko programowania, to należałoby uczyć go w języku innym niż BASIC. Powstaje zatem pytanie — w jakim? Pozostawiam to pytanie na razie bez odpowiedzi, bowiem jest jeszcze drugi problem, który moim zdaniem jest jeszcze ważniejszy od pierwszego, chodzi tu o pierwsze zetknięcie ucznia z informatyką przez programowanie. Nawet przy założeniu, że programowanie odbywać się będzie w najlepszym z możliwych języku programowania, pierwsze zetknięcie z informatyką przez programowanie może łatwo doprowadzić do stworzenia obrazu, iż informatyka to w gruncie rzeczy tylko programowanie lub znajomość komputera. Stąd już tylko krok do wniosku, że jeśli nauczę się pisać programy w języku BASIC i jeśli będę umiał posługiwać się mikrokomputerem SPECTRUM, to w zasadzie z informatyką jestem za pan brat. Jest to obraz nie tylko fałszywy, ale także wielce szkodliwy dla szerokiej opinii i dla tych, którzy w przyszłości będą zajmować się informatyką, bądź zechcą korzystać z jej narzędzi, przy rozwiązywaniu autentycznych problemów. Nasuwa się zatem następujący wniosek (ale nie — postulat) — należy stanowczo przeciwstawiać się temu, aby uczniowie, a co za tym idzie, coraz szersze kręgi społeczeństwa, pojmowały informatykę jako programowanie lub czy jako znajomość tego czy innego komputera.

Dochodzimy więc do zasadniczego pytania — czym jest informatyka na użytek dydaktyki, a przede wszystkim — dydaktyki szkolnej.

Najwłaściwsza wydaje się koncepcja określenia informatyki jako nauki o rozwiązywaniu problemów za pomocą komputera. Główną osią tematyczną tej koncepcji jest proces rozwiązywania problemu za pomocą komputera, składający się z następujących czterech etapów:

1. **Formułowanie problemu.**
2. **Poszukiwanie metody rozwiązania problemu.**
3. **Tworzenie algorytmu.**
4. **Realizacja algorytmu.**

Wróćmy zatem do sytuacji wyjściowej, gdy mamy dziecko i mikrokomputer. Kontakt dziecka z mikrokomputerem, a więc realizacja jakiegoś zadania, powinien stawać się kontaktem z informatyką, a więc powinien być realizowany w ramach przedstawionego schematu rozwiązywania problemu z pomocą komputera. Takie podejście do realizacji zadania będzie na ogół w jakimś stopniu uproszczone, ale jedno jest tutaj ważne, aby to podejście zawsze różniło się istotnie od podejścia następującego: chcę uzyskać nową grę, więc siadam i zaczynam pisać program.

Dziecko, a właściwie uczeń, oczywiście po tej drodze nie może pójść sam — potrzebuje przewodnika, opiekuna. Tym najwłaściwszym przewodnikiem i najlepszym opiekunem powinien być nauczyciel. A tymczasem, wszystko wskazuje na to, że w tym wirze mikrokomputerowym, jaki nas ogarnia, o nauczycielu nikt nie pamięta. Kluby mikrokomputerowe okupowane są głównie przez dzieci i młodzież. Sprzęt w znacznej części też znajduje się w rękach dzieci i młodzieży. A ilu nauczycieli ma własne mikrokomputery? Osobiście znam tylko jeden taki przypadek. O programach już była mowa. Kursy programowania w BASICU lub asemblerze też, o ile wiem, organizuje się głównie dla dzieci i młodzieży.

Myślę, że ten stan, który analizujemy, dobrze oddaje następujący, z życia wzięty obrazek. Mój kolega, matematyk, zwrócił się kilka dni temu do mnie z prośbą — pytaniem, czy ma kupić swojemu synowi w wieku 11 lat, ZX SPECTRUM. Szybko doszliśmy do wniosku, że oczywiście tak. I wtedy skierował do mnie drugą prośbę: — jeśli tak, to proszę cię, podaj mi kilka argumentów, abym mógł jeszcze przekonać swoją żonę, bo ona jest temu przeciwna. Zdziwiłem się i pytam: — a co robi twoja żona, przecież jest nauczycielką? Tak — odparł — rzeczywiście jest nauczycielką matematyki w szkole średniej. Zdenerwowałem się, ale po namyśle powiedziałem: — powiedz jej po prostu, że te mikrokomputery nie są zabawkami dla dzieci, ale powinny być i wkrótce będą niezbędnymi narzędziami pracy nauczyciela, w szczególności nauczyciela matematyki.

Niestety, nie jest to historia zmyślona.

Sądzę, że z powyższego wyraźnie widać, że pierwszoplanowym zadaniem przed jakim stoimy jest problem nie ucznia, nie mikrokomputera, lecz problem nauczyciela. Sądzę także, iż widać, jak problem ten powinien być rozwiązywany. Ale zanim do tego przejdziemy, jeszcze kilka uwag.

Wydaje mi się, iż to co obecnie dzieje się w rozpatrywanym przez nas układzie prowadzi w ślepy zaułek. Mikrokomputer jest przede wszystkim dla dzieci, ale niestety wprowadzamy te dzieci w programowanie w BASICU lub w asemblerze, bez zasadniczego przygotowania. Nauczyciel nie ma prawie żadnego kontaktu z mikrokomputerem i na ogół nie zna informatyki. Czego można więc spodziewać się w efekcie? Dzieci będą oczywiście dochodzić do małpiej wprawy w posługiwaniu się mikrokomputerem, co w skutkach, o dziwo, może okazać się jednak bardzo szkodliwe. Nauczyciel zaś, z wielu powodów będzie stronić od mikrokomputera.

Wnioski nasuwają się same, spróbujmy je więc uporządkować.

Na studiach nauczycielskich na matematyce, skąd w dużej mierze rekrutują się przyszli nauczyciele informatyki, konieczne jest wprowadzenie wykładu z informatyki w wymiarze co najmniej dwóch semestrów. Laboratorium informatyczne powinno też odbywać się co najmniej przez dwa semestry i konieczne jest wyposażenie go przynajmniej w 4—6 mikrokomputerów typu ZX SPECTRUM.

Podobnie na studiach podyplomowych dla nauczycieli, prowadzonych przy wydziałach matematyki uniwersytetów, należy zwiększyć liczbę zajęć z informatyki do podobnego wymiaru.

Musimy też zatroszczyć się o nauczycieli nie podejmujących studiów podyplomowych, obejmując ich opieką merytoryczną w zakresie informatyki, organizując dla nich na — przykład — kluby mikrokomputerowe. Podkreślam to raz jeszcze, że chodzi tu o nauczycieli, a nie o uczniów. W klubach tych uczono by informatycznego podejścia do rozwiązywania problemów z pomocą komputera. Bardzo ważnym zadaniem tych klubów byłoby przekonanie nauczyciela, iż mikrokomputer powinien być przez niego traktowany przede wszystkim jako nowe, potężne narzędzie pracy.

Należy również podjąć systematyczne prace nad oprogramowaniem mikrokomputerów z myślą o nauczycielu. Chodzi tu o tworzenie takich programów, które służyłyby przede wszystkim nauczycielowi, aby uwierzył on w mikrokomputer i przekonał się, że nie jest to tylko zabawka dla dzieci. Nie miejsce tu na szersze rozwijanie tematu, ale podam chociaż dwie propozycje rozwiązania tego zadania. Pierwsza dotyczy nauczania początkowego, a nawet przedszkolnego, i wiąże się z nauką czytania. Okazuje się, że nasze dzieci źle czytają. A przy użyciu mikrokomputera nauczyciel będzie mógł im znacznie pomóc. Ten projekt jest jeszcze w stadium pomysłu. Natomiast na ukończeniu jest projekt dotyczący generowania i rozwiązywania krzyżówek. Takich zwykłych krzyżówek, jakie rozwiązujemy w gazetach, czy w magazynie „Relaks". Taka krzyżówka może być atrakcyjną formą dla zwykłego testu [1]. Z uporem wysuwam tutaj zawsze na plan pierwszy nauczyciela. Ale nie obawiajmy się, że przez to ucierpi uczeń. Te wszystkie przedsięwzięcia służą przecież w efekcie nie tylko nauczycielowi, lecz także, a może i przede wszystkim uczniowi.

Dopiero odpowiednio przygotowany i wyposażony nauczyciel może stać się oczekiwanym opiekunem i przewodnikiem ucznia. Może pojawić się z mikrokomputerem na lekcji, może poprowadzić kółko zainteresowań, może wreszcie podjąć się prowadzenia przedmiotu „Informatyka". Ale zanim nauczyciel matematyki podejmie się prowadzenia tego przedmiotu, może przygotowywać się do tego przedmiotu w ramach lekcji matematyki. Nie rozwijam tutaj tego wątku, ponieważ zagadnienie to omówiłem w czasopiśmie „Matematyka" [2], do którego odsyłam zainteresowanych.

Na zakończenie chciałbym jeszcze raz powrócić do sprawy języka BASIC. W przedstawionym nowym podejściu mamy w zasadzie do czynienia już z innym problemem, a mianowicie problemem języka programowania na użytek dydaktyki. Myślę, że bardzo by się tu przydało pełne i rzetelne studium na temat języka BASIC i to głównie pod kątem jego użyteczności, czy może szkodliwości dla ucznia, nauczyciela, a także użytkownika, który po raz pierwszy styka się z językiem programowania. Chodzi tu nie tyle o to, aby ten język, oprócz złej opinii, doczekał się jeszcze formalnego wyklęcia, lecz może bardziej o to, że takie studium może być pomocne w zrozumieniu tego, jakimi cechami powinien się charakteryzować język programowania wykorzystywany w celach dydaktycznych, lub też jak prawidłowo wykorzystywać istniejące rozwiązania.

EMANUEL CZYŻO

LITERATURA

[1] Borowiecki M., Czyżo E.: Krzyżówka w szkole (w przygotowaniu)
[2] Czyżo E.: Przygotowanie do nauczania informatyki. Matematyka, 6, 1983.

Odpowiedź na zagadkę ze str. 28

Zamieszczony program podczas wykonywania na komputerach ICL, CDC lub IBM — rozpoznaje właściwy typ maszyny wprowadzając na ten temat odpowiednią informację. Proponuję Czytelnikom rozszerzenie tego programu o rozpoznawanie innych typów komputerów, a raczej kompilatorów języka FORTRAN.

III/B. Metody sortowania wewnętrznego

Samotesty, opracowane i opublikowane przez Association for Computing Machinery, nie stanowią standardu egzaminacyjnego i są przeznaczone wyłącznie do samodzielnego sprawdzania własnej wiedzy [1].

PYTANIA

Pytania samotestu III/B są związane z sześcioma najszerzej znanymi metodami sortowania wewnętrznego. Ponieważ jednak w piśmiennictwie fachowym panuje duża dowolność i niejednoznaczność nazewnictwa tych metod — zostały one na potrzeby niniejszego samotestu zdefiniowane roboczo, przez wymienienie w tabeli ich najważniejszych właściwości [2]. Podano także najczęściej stosowane nazwy angielskie, oraz zaproponowano terminologię polską.

1. Czas potrzebny do posortowania n rekordów, zawierających pole kluczowe o długości x cyfr — przy zastosowaniu metody A będzie proporcjonalny do:

   a) n
   b) k
   c) n+k
   d) n*log(k)

2. Które z poniższych stwierdzeń jest najbardziej trafne, jeżeli zastosujemy metodę B do posortowania 10 rekordów?

   a) posortowanie wymagać będzie 100 porównań i 10 przestawień
   b) posortowanie wymagać będzie 81 porównań i 9 przestawień
   c) w pewnych sytuacjach wcale nie muszą wystąpić przestawienia
   d) czas porównywania ulegnie zmniejszeniu, jeżeli rekordy będą posortowane

3. Jakie z poniższych określeń jest najmniej trafne, jeśli dla posortowania 10 rekordów użyliśmy metody C?

   a) wystąpi tylko 9 porównań, bez przestawień
   b) potrzeba będzie 45 porównań i 45 przestawień
   c) metoda ta jest szybsza dla zbioru nie uporządkowanego (o przypadkowym uporządkowaniu), niż dla zbioru o uporządkowaniu odwrotnym do wymaganego
   d) czas porównywania będzie krótszy, jeżeli początkowe rekordy zbioru będą uporządkowane
   e) czas przestawiania jest niezależny od długości rekordu

4. Załóżmy, że dla posortowania zbioru stosujemy metodę D. Które z pierwotnych uporządkowań zbioru spowoduje, że czas sortowania będzie najkrótszy?

   a) zbiór jest uporządkowany
   b) zbiór do sortowania jest uporządkowany odwrotnie
   c) uporządkowanie pierwotne zbioru jest przypadkowe
   d) pierwotne uporządkowanie zbioru nie ma w tym przypadku żadnego znaczenia

5. Które z poniższych określeń jest najmniej trafne, jeśli dla posortowania pewnego zbioru zastosowano metodę E?

   a) operacje dzielenia i łączenia wykonywane są raczej poprzez operacje na indeksach (ang. judicious indexing), aniżeli przez przemieszczanie rekordów
   b) grupy rekordów o takich samych w przybliżeniu rozmiarach mogą być sortowane równocześnie
   c) metoda ta może być stosowana jedynie w przypadku, gdy liczba rekordów jest potęgą dwójki
   d) metoda ta jest wolniejsza od metody C, w przypadku gdy rekordy początkowe występują we właściwym uporządkowaniu

6. Przyjmijmy, że sortujemy n rekordów przy zastosowaniu metody F. Przy założeniu, że najpierw jest przetwarzany najkrótszy segment uzyskany z podziału, jaka jest maksymalna liczba segmentów dłuższych niż jeden rekord, które muszą być brane pod uwagę w dowolnym momencie?

   a) 3
   b) n
   c) n-1
   d) n/2
   e) $\log_2(n)$

7. W której z poniższych metod wystąpi najmniejsze wykorzystanie pierwotnego układu rekordów, których pola kluczowe są identyczne?

   a) A
   b) B
   c) D
   d) F

8. Załóżmy, że mamy półkę biblioteczną o długości 3,5 m pełną książek. Kilka książek zostało przez wypożyczających przestawionych i źle ustawione książki stanowią ok. 0,7 metra długości. Którą z metod zastosowałbyś dla uporządkowania książek?

   a) A
   b) C
   c) D
   d) F

9. Niektóre programy sortujące nazywa się porównującymi, ponieważ w sposób bezpośredni dokonują one porównywania zawartości kluczy rekordów sortowanych. Która z poniższych metod ma najmniej cech metody porównującej?

   a) A
   b) B
   c) E
   d) F

10. Niektóre programy sortujące nazywa się rozdzielającymi (ang. distributive), ponieważ rozdzielają one rekordy pomiędzy podzbiory, z których jeden zawiera rekordy o wartości kluczy wszystkich rekordów drugiego podzbioru. Która z poniższych metod ma najmniej cech metody rozdzielającej?

    a) D
    b) E
    c) F

11. Niektóre metody sortowania wymagają dodatkowego obszaru pamięci dla zapamiętania wskaźników lub kopii sortowanych rekordów. Która z niżej wymienionych metod może zostać uznana za mającą najmniejsze zapotrzebowanie na dodatkową pamięć?

    a) A
    b) B
    c) D
    d) F

12. Niektóre metody sortowania nie wymagają dostępu do ostatnich rekordów zbioru sortowanego, aż do momentu zbliżenia się do końca procesu sortowania. Inne metody nie wymagają dostępu do rekordów sortowanych na początku, od chwili ich posortowania. Które z metod wymienionych poniżej wymagają stałego dostępu do wszystkich rekordów zbioru przez cały czas trwania procesu sortowania?

    a) B
    b) C
    c) D
    d) E

ROZWIĄZANIA

1 — d, 2 — c, 3 — e, 4 — c, 5 — c, 6 — c, 7 — d, 8 — b, 9 — a, 10 — b, 11 — b, 12 — d

[1]) Niniejszy samotest zamieszczony został w „Communications of the ACM vol. 20 (No. 9, September 1977) str. 621—624.

Podstawowe metody sortowania wewnętrznego

Metoda	Krótka charakterystyka	Spotykane nazwy
A	Umieszczaj kolejne rekordy w wydzielonych obszarach odpowiadających wszystkim możliwym wartościom najmłodszej pozycji klucza; połącz obszary i ponów postępowanie dla najbliższej starszej pozycji, aż do wyczerpania całego klucza	— metoda najmłodszej pozycji (ang. up ward radix sort) — metoda cyfrowa (ang. digital sort)
B	Odszukaj rekord o najmniejszej wartości klucza i umieść go przed pierwszym nie posortowanym rekordem zbioru; kontynuuj postępowanie aż wszystkie rekordy zostaną ustawione według rosnących wartości kluczy	— metoda kolejnych najmniejszych wartości (ang. successive minimal sort) — metoda bezpośredniej selekcji (gng. straight selection sort) — metoda zamiany i wyboru (ang. selection and exchange sort)
C	Wstaw kolejny rekord we właściwe miejsce już przetworzonej części zbioru, aż do wyczerpania nie przetworzonych jeszcze rekordów	— metoda wpasowywania (ang. insertion sort) — metoda wartościowania (ang. ranking sort)
D	Dowiąż kolejny rekord do właściwego wolnego węzła struktury drzewiastej, zgodnie z powinowactwem na poszczególnych poziomach; po umieszczeniu wszystkich rekordów w strukturze drzewiastej — kopiuj jej uporządkowane rekordy do nowego obszaru pamięci	— metoda powinowactwa (ang. ancestral sort) — metoda dychotomiczna (ang. binary distributive sort) — metoda wszczepiania (ang. tree insertion sort)
E	Podziel nie przetworzony zbiór na wiele małych grup; sortuj rekordy w każdej takiej grupie z osobna (stosując jedną z powyższych metod, np. C); łącz po dwie grupy w większą i ponów sortowanie; postępuj analogicznie z kolejnymi grupami, aż do uzyskania jednej uporządkowanej grupy wszystkich rekordów	— metoda Schella (ang. Schell sort) — metoda przesiewania (ang. sifting sort) — metoda zanikających inkrementów (ang. diminsning increment sort)
F	Podziel nie przetworzone rekordy na trzy segmenty: jednorekordowy segment środkowy oraz dwa segmenty rekordów odpowiednio młodszych i nie; każdy segment wielorekordowy w taki sam sposób dalej, aż wszystkie segmenty będą zawierać już tylko po jednym rekordzie	— metoda trójsegmentowa (ang. guick sort) — metoda podziału i wymiany (ang. partition and exange sort)

[2]) Podane w tabeli opisy metod sortowania prezentuję, zgodnie z jej tytułem, zasadę ogólną. Tłumaczenie najczęściej spotykanych nazw metod nie zawsze jest szczęśliwe, ale też nie ma tutaj uznanych wzorców. W polskiej literaturze informatycznej mamy w zasadzie trzy pozycje zawierające opisy różnych metod sortowania wewnętrznego.

[1] Turski W. M.: Struktury danych. WNT, Warszawa, 1976 Zawiera opisy i analizę siedmiu metod sortowania nazwanych po polsku (niestety, bez podania oryginalnych nazw angielskich)

[2] Wirth N.: Algorytmy + struktury danych = programy. WNT, Warszawa, 1980.
Jest to tłumaczenie z języka angielskiego, zawierające trzynaście różnych metod sortowania zapisanych w postaci procedur w języku PASCAL wraz z wyczerpującym omówieniem. Niestety i tutaj wprowadzono tylko polskie nazwy metod. Należy też zwrócić uwagę na różnice w nazewnictwie między [1] a [2] (np.: tam „metoda włoczenia", a tu „sortowanie bąbelkowe").

[3] Aho A. V., Hopcroft J. E., Ullman J. D.: Projektowanie i analiza algorytmów komputerowych. PWN, Warszawa, 1983.
Jest to również tłumaczenie z języka angielskiego zawierające opisy pięciu podstawowych metod sortowania, wyczerpująco przeanalizowanych pod kątem złożoności obliczeniowej.
Kilka metod sortowania zostało też opisanych w INFORMATYCE, w dziale ALGORYTMY (nr 4—5/82, 1/83). Przyp. red.

KOMENTARZE

Ad 1. Por. metoda A.
Ad 2. Por. metoda B. Jeżeli rekordy są już odpowiednio uporządkowane, nie są potrzebne przestawienia.
Ad 3. Por. metoda C. Dla zbiorów o długich rekordach całkowity czas przestawiania może zostać zmniejszony w drodze manipulowania odsyłaczami, zamiast samymi rekordami; [2] (str. 155); [5] (str. 168); [7] (str. 6).
Ad 4. Por. metoda D.
Ad 5. Por. metoda E.
Ad 6. Por. metoda F.
Ad 7. [5] (str. 169); [7] str. 95).
Ad 8. [5] (str. 170).
Ad 9. Program sortujący metodą najmłodszej pozycji (ang. upward radix sort) nie porównuje ze sobą kluczy rekordów, lecz umieszcza rekord w zależności od wartości jego klucza.
Ad 10. [5] (str. 154—155).
Ad 11. [7] (str. 95).
Ad 12. Program sortujący metodą Shella porównuje każdy rekord z innym co najmniej raz za każdym przebiegiem.

Poszczególne metody opisano:

A. [3] str. 179—180; [4] str. 148—150; [5] str. 155—156; [7] str. 16—18
B. [2] str. 154; [3] str. 139—142; [4] str. 10—12; [6] str. 1300; [7] str. 50—60
C. [1] str. 38—54; [2] str. 155; [3] str. 80—84, 95—99; [4] str. 32—35; [5] str. 148—149; [7] str. 60—68; [8] str. 93
D. [3] str. 422—428; [4] str. 59—86; [5] str. 157—158; [7] str. 34—39; [8] str. 94—95
E. [3] str. 84—92; [4] str. 37—43; [5] str. 149—150; [7] str. 70—77; [8] str. 93—94
F. [3] str. 114—123; [4] str. 97—107; [5] str. 151—152; [7] str. 77—92; [8] str. 95—96.

LITERATURA

[1] Flores I.: Computer Sorting. Prentice-Hall, Englewood Cliffs., New York, 1969
[2] Hull T. E., Day Dawid D. F.: Computers and Problem Solwing Addison-Weseley (Canada), Don Mills, Ontario, 1970
[3] Knuth D. E.: The Art of Computer Programming, vol. 3: Sorting and Searching. Addison-Wesley, Reading, Mass., 1975
[4] Lorin H.: Sorting and Sort Systems. Addison-Wesley, Reading, Mass., 1975
[5] Martin W. A.: Sorting. Computing Surveys, vol. 3, No. 4 December 1971
[6] Ralston A., Meek Ch. L.: Encyclopedia of Computer Science. Pertocelli Charter, New York, 1976
[7] Rich R. P.: Internal Sorting Methods Illustrated with PL/1 Programms. Prentice-Hall, Englewood Cliffs., New York, 1972
[8] Schember K. A., Rumsey J. R.: Minimal Storage Sorting and Searching Techniques for RAM Applications: A Tutorial Computer, June 1977.

Adaptowali z angielskiego

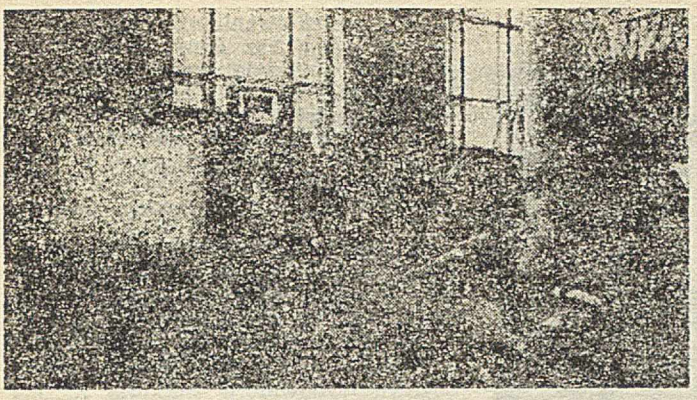
ADAM B. EMPACHER
LUDWIK J. ROSSOWSKI

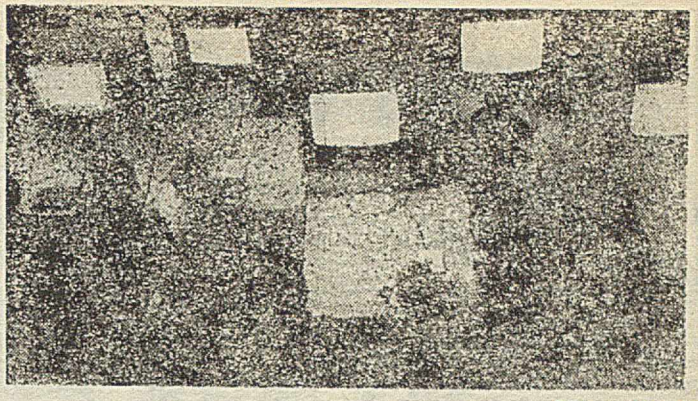
Ogłoszenia • Ogłoszenia • Ogłoszenia • Ogłoszenia

Zakupimy wycofane elementy jc. ODRA 1204/1304 lub zamienimy na mikrokomputer „Spectrum" lub „Amstrad". NAPRAWA MULTIMETRÓW V527/V640, Warszawa, tel. 47-22-57, 20-90-61 w. 93.

EO/891/K/85

Ogłoszenia • Ogłoszenia • Ogłoszenia • Ogłoszenia





Fot. KACPER

Mikrokomputery w Pałacu Młodzieży

Na początku grudnia ub.r. Pałac Młodzieży w Warszawie przeżył inwazję odwiedzających. Na zorganizowaną przez Pracownię Informatyczną imprezę „Mikrokomputer dla każdego" — pokaz działania dwunastu mikrokomputerów (będących w większości prywatną własnością członków) — przybyło ponad cztery tysiące osób. Potem w kolejnej imprezie, promowanej przez firmy POLBRIT i UNIMOR (2—10 stycznia br.), uczestniczyły już zorganizowane grupy dzieci — tak, aby każde z nich mogło choćby przez kilkanaście minut oswajać się z mikrokomputerem ZX SPECTRUM, przez samodzielny udział w grze komputerowej.

Jak do tego doszło? Otóż dowiedziano się niedawno, że w pewnej szkole w Warszawie stoi nie używany minikomputer MERA 300, którym przez kilka lat nikt się nie zajmował. Udało się przekazać go do Pałacu Młodzieży, a przy okazji odkryto następną nie używaną MERĘ — dzięki temu złożono jedną z dwóch i przed rokiem powstała Pracownia Informatyczna. Wkrótce przybył mikrokomputer ZX81, uzyskany dzięki pomocy Leszka Wilka, prezesa klubu ABAKUS. Jakim sposobem udało się zdobyć ZX SPECTRUM, tego nie wiem. Przecież jego cena oznacza już zakup inwestycyjny, a to trzeba ująć w planach i wykonać tak wiele biurokratycznych czynności, by wreszcie po opinii rzeczoznawcy, któremu trzeba przystępnie wytłumaczyć czym jest mikrokomputer, i — po uzyskaniu zezwolenia Wydziału Handlu...

Dwa mikrokomputery i MERA 300 — to cały sprzęt będący własnością Pracowni. W czasie mojej wizyty działało tylko SPECTRUM. MERA 300 była unieruchomiona na stałe. Mikrokomputerowi ZX81 brakowało natomiast pamięci; od dwóch miesięcy była ona w naprawie — cóż, firma, do której ją wysłano, ma z pewnością pilniejsze zadania. Całe szczęście, że młodzież przychodząca na zajęcia, a także instruktorzy, przynoszą własny sprzęt.

Dzieci biorących regularny udział w zajęciach jest około setki, głównie — z końcowych klas szkoły podstawowej i ze szkół średnich. Obecnie są podzieleni na trzy grupy: początkującą i zaawansowaną w programowaniu oraz sprzętową. Zajęcia odbywają się w formie wykładów i ćwiczeń praktycznych.

Grupa początkująca uczy się BASICA, przechodząc stopniowo do samodzielnego przygotowania i pisania programów. Choć najbardziej interesują ich gry komputerowe, zakłada się, że nie będzie to zajęcie podstawowe. Przede wszystkim mają oni uruchamiać programy, choćby do gier.

W grupie zaawansowanej nie było kłopotów z opanowaniem języka asemblera Z80, a FORTH uznano za łatwiejszy i lepiej przyswajalny od PASCALA.

Grupa sprzętowa ma do wykonania skromne zadania, m.in. sprzężenie drukarki DZM z komputerem ZX81, ale też ambitniejsze plany, w tym — zbudowanie własnego mikrokomputera.

Pracownia nie ma własnego pomieszczenia, nie ma dostatecznej ilości sprzętu i dostępu do części, które są obecnie kupowane kanałami prywatnymi, nie ma dostępu do literatury ani możliwości jej powielania (nawet zdobycie instrukcji dla ZX81 po polsku jest problemem, a zamówienie na prenumeratę INFORMATYKI bibliotekarka wysłała nie tam gdzie trzeba), a także — co najważniejsze — w ogóle nie zatrudnia informatyka, który służyłby młodzieży rzetelną wiedzą. Instruktorami są: dyplomant Wydziału Mechaniki Precyzyjnej i absolwent Wydziału Elektrycznego Politechniki Warszawskiej. Obaj są wychowankami Pałacu Młodzieży — na zajęcia różnego rodzaju uczęszczali tu od dziecka. Teraz poświęcają swoim wychowankom ok. 10 godzin tygodniowo.

Kto pomoże w wyposażeniu Pracowni? Może Polskie Towarzystwo Informatyczne, deklarujące pewne sumy na działalność mikrokomputerową? Może firma POLBRIT, współorganizująca styczniową imprezę? A może Ministerstwo Oświaty i Wychowania, rozdzielające w owym czasie sprzęt mikrokomputerowy (300 egzemplarzy ZX SPECTRUM, podarowanych przez Reitera) oraz środki finansowe? Czy Zrzeszenie MERA..., choć jego MERITUM-I uzyskało wstępnie dość niską ocenę tej młodzieży. Jak na razie, pierwsze kontakty z wymienionymi instytucjami pchnęły jedynie młodzież w objęcia biurokracji.

Wizyta w Pałacu Młodzieży prowokuje do jednego tylko generalnego wniosku. Jeżeli nie podejmiemy odpowiednich działań dostatecznie wcześnie, to tego żywiołowego ruchu nic nie będzie w stanie opanować. Programowania w BASICU i chwytów programowych w języku asemblera będziemy uczyć się od naszych dzieci, niestety — samouków.

JANUSZ ZALEWSKI

Stały kontakt z INFORMATYKĄ gwarantuje tylko prenumerata

Do 10 listopada można wpłacać na 1986 rok.
Lepiej nie czekać z decyzją — wszystkie zamówienia zostaną zrealizowane!

Rozproszone systemy przetwarzania danych

W dniach 10—15 września 1984 r. odbyła się konferencja naukowa zorganizowana przez Politechnikę Wrocławską, przy współudziale zespołów naukowo-badawczych uczestniczących w realizacji zadań objętych Problemem resortowym pn. „Rozwój komputeryzacji szkół wyższych". Konferencja ta była kontynuacją dwóch imprez: odbywającej się po raz siódmy szkoły naukowej „Sieci komputerowe i systemy rozproszonego przewarzania danych" oraz organizowanego corocznie (od 1979 r. wspólnie z Politechniką w Magdeburgu — NRD) międzynarodowego seminarium „Banki danych".

W konferencji wzięło udział 65 uczestników, w tym 13 z zagranicy (ZSRR, RFN, CSRS, Wielkiej Brytanii i NRD). Krajowi uczestnicy reprezentowali szkoły wyższe z Warszawy, Poznania, Krakowa, Torunia, Katowic, Zielonej Góry i Wrocławia oraz inne ośrodki naukowo-badawcze w kraju.

Wygłoszono 41 referatów, w tym 12 wykładów monograficznych i 29 komunikatów. Obrady prowadzone były w sesjach ogólnych, dla wszystkich uczestników konferencji, oraz w dwóch równolegle odbywających się sesjach problemowych: („Banki danych") oraz („Sieci komputerowe").

Wykłady obejmowały głównie zagadnienia metodyczne informatyki, niezależnie od zmian zachodzących w technologii sprzętu komputerowego oraz sposobów łączenia go w sieci. Sposób pojmowania informatyki, jako dyscypliny naukowej, podjęto w czterech wykładach poświęconych filozoficznym podstawom inżynierii informatycznej, logicznej organizacji zbiorów danych w systemach rozproszonych, synchronizacji w systemach rozproszonych baz danych oraz semantyce baz danych w aspekcie rozproszonego przetwarzania. Inna grupa wykładów obejmowała problematykę projektowania architektury systemów rozproszonego przetwarznia danych oraz metod projektowania sieci komputerowych i lokalnych sieci komputerowych. Przeprowadzono także analizę zalet i wad obecnie stosowanych technik dostępu do zasobów sieci lokalnych oraz porównanie oczekiwań projektowych z doświadczeniami eksploatacyjnymi sieci typu WAN i LAN, użytkowanych w uniwersytetach brytyjskich. Tematyka ostatniego z referatów stanowi kontynuację zajęć dydaktycznych zapoczątkowanych w 1980 r. w ramach Szkoły „Sieci Komputerowe '80".

W innych referatach zaprezentowano problemy optymalizacji projektowania i eksploatacji sieci komputerowych, zastosowanie modeli symulacyjnych i eksperymentalnych w komputerowo wspomaganym projektowaniu systemów informatycznych oraz dwuwarstwowy system obsługi.

Podjęto również tematykę klasyfikacji oprogramowania użytkowego. Zaprezentowano system EPOS, opracowany dla Europejskiej Organizacji Udostępniania Oprogramowania i Przekazywania Informacji (EASIT) oraz użytkowany w sieci EURONET. Podstawową własnością systemu EPOS jest oparcie klasyfikacji oprogramowania na podobieństwie zastosowań, modeli formalnych i metod rozwiązywania.

Tematyka komunikatów szkoły „Sieci komputerowe" obejmowała doniesienia ilustrujące wyniki zakończonych badań, rozwiązań projektowych i wdrożeń, związanych z projektowaniem i budową sieci komputerowej MSK/ASK (Międzyuczelniana Sieć Komputerowa; Akademicka Sieć Komputerowa) realizowanej w Problemie resortowym.

W komunikatach wygłoszonych na posiedzeniach seminarium „Banki danych" prezentowano zagadnienia związane z projektowaniem, implementacją i eksploatacją baz danych. Większość referatów dotyczyła wzrostu efektywności eksploatacji baz danych oraz metod ich modelowania i oceny. W innej grupie przedstawiono zagadnienia projektowania baz rozproszonych, w tym języka dostępu do zasobów, oraz tematykę administrowania bazami danych.

Wszystkie wykłady oraz wybrane komunikaty ukażą się w serii „Biblioteka Informatyki Szkół Wyższych" oraz w wydawnictwie „Problemy projektowania, wdrażania i eksploatacji baz danych. Część II". Części pierwsze obydwu wydawnictw znajdują się już w sprzedaży.

W treści komunikatów dominowały zagadnienia o charakterze eksperymentalnym i praktycznym. Wskazuje to na postęp, w stosunku do treści komunikatów wygłoszonych podczas poprzedniej Szkoły, wśród których przeważały prace o charakterze koncepcyjnym. Świadczy także o zmianach zachodzących w ośrodkach badawczych, polegających na wzroście doświadczeń z zakresu projektowania i wdrażania systemów rozproszonego przetwarzania danych.

W dyskusjach prowadzonych w trakcie konferencji wskazywano na:

- konieczność udoskonalenia aparatu badawczego stosowanego w informatyce, pod kątem ułatwienia weryfikacji rozwiązań projektowych systemów informatycznych,
- znaczenie oceny w projektowaniu systemów informatycznych z punktu widzenia ich architektury.

Należy podkreślić, że omówiona konferencja, jest jedną z nielicznych imprez międzynarodowych organizowanych w kraju, a także w krajach socjalistycznych, na temat sieci komputerowych i systemów rozproszonego przetwarzania danych.

JULIAN DĘBOWY
Politechnika Wrocławska

Konferencja FTSD'85

Systemy tolerujące uszkodzenia i diagnostyka

VIII Międzynarodowa Konferencja FTSD'85 odbyła się w dniach 10—12 września 1985 r. w ośrodku wypoczynkowym PBSz., w pensjonacie SILESIANA w Lubińcu-KOKOTKU (k. Katowic).

Jej organizatorami były: Politechnika Śląska w Gliwicach, Instytut Maszyn Matematycznych w Warszawie, Instytut Systemów Sterowania w Katowicach i Uniwersytet Śląski w Katowicach.

Program konferencji obejmował następujące sesje:

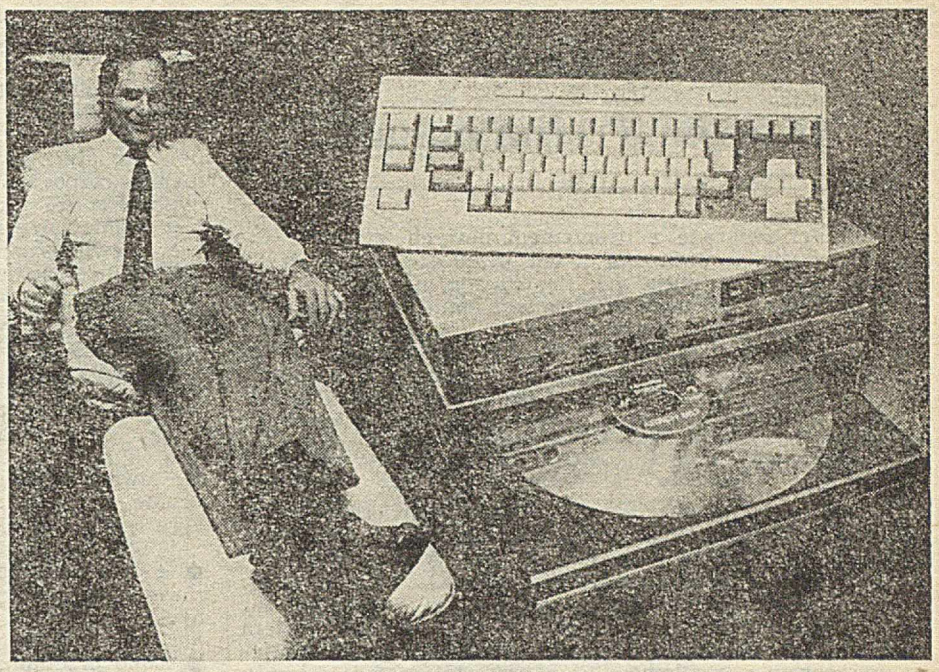
- **Plenarną,**
- **Projektowania systemów tolerujących uszkodzenia,**
- **Testowania i diagnostyki,**
- **Układów samosprawdzających się i kodów korekcyjnych,**
- **Niezawodności i testowania sieci komputerowych**
- **Niezawodności oprogramowania,**
- **Urządzeń testujących.**

Językami konferencji były: język angielski, polski i rosyjski.

Szczegółowych informacji udziela przewodniczący Komitetu Organizacyjnego:

dr Marian Budka
Politechnika Śląska
ul. Pstrowskiego 16, 44-100 Gliwice
tel. 31-56-60 w. 220. teleks 036304

Dyski laserowe



Następnym mutantem japońskiej rodziny mikrokomputerów jest model PX-7 firmy PIONEER. Jest to już kolejny przykład połączenia mikrokomputerów z coraz bardziej popularnymi urządzeniami rozrywkowymi. Po modelu CX-5M firmy YAMAHA przystosowanym do natychmiastowej współpracy z syntezatorem (wykokorzystującym cyfrową obróbkę sygnałów i modulację częstotliwości) pojawia się następne zastosowanie — dyski laserowe LD-700. Również firma JVC oferuje już trzeci mikrokomputer z serii MSX do współpracy z alternatywnymi dyskami systemu VHD. Podobne plany ma firma PHILIPS — mikrokomputer MSX już jest na rynku — obecnie pracuje się nad jego połączeniem z dyskami systemu LASERVISION.

Co jest efektem takiego połączenia: na ekranie nałożone są na siebie dwa obrazy — z komputera i z dysku. Obraz z dysku jest bardzo dobrej jakości i dużo bardziej realistyczny niż komputerowy, natomiast obraz z komputera to wszelkie napisy i sterowanie. W zależności od wysterowania i przebiegu gry, zmieniają się obrazy z dysku (czas wyboru dowolnego obrazu z 54 000 na każdej stronie nie przekracza maksymalnie 6 s.). Opóźnienie to nie pozwala na zastosowanie dysków laserowych do takich gier, jak wyścigi, różnego rodzaju strzelaniny, ale pozostaje tak szerokie pole zastosowań, jak edukacja, symulacja i... inne przygody.

Pojemność dysku jest wprost przeogromna. Dla porównania na obu stronach można zmieścić co najmniej trzy encyklopedie tej wielkości co Encyclopedia Americana lub Brittanica, z niemal natychmiastowym dostępem do wizualnej informacji.

Dotychczasową wadą tych dysków z punktu widzenia komputerów jest niemożliwość powtórnego zapisu, choć prace wciąż trwają i należy się spodziewać przełamania i tej bariery.

Na zakończenie coś dla snobów — można do zestawu: komputer — dysk laserowy dokupić specjalny, bardzo wygodny, fotel na którym można odczuwać efekty trzęsienia ziemi o różnym natężeniu.

Opracował M. G.

na podstawie
YOUR COMPUTER nr 12/84

Dyski 3M

MINNESOTA MINNING and MANUFACTURING COMPANY — znany producent urządzeń dyskowych — wprowadza na rynek nowy model dysku dla mikrokomputerów. Dysk ten produkowany jest w oparciu o tzw. technologię zapisu na rozciągniętej powierzchni (ang. stretched surface data recording). Wytwarzany w ten sposób dysk o średnicy 5¼ cala mieści 20-krotnie więcej danych niż tradycyjny dysk sztywny. Dyski produkowane w oparciu o nową technologię wymagają specjalnych urządzeń odczytujących.

Dysk taki powstaje w efekcie jednoczesnego naciągnięcia elastycznego materiału magnetycznego na obie strony sztywnego, plastikowego płaskiego dysku. Mieści on aż 345 ścieżek danych (standardowy 5¼-calowy dysk elastyczny zawiera zwykle tylko 48 ścieżek). Nowe dyski eliminują dotychczasową podstawową wadę dysków elastycznych, jaką są błędy powstające podczas przechowywania, w efekcie odkształceń powodowanych wysoką temperaturą lub nadmierną wilgotnością powietrza. Powłoka magnetyczna wytwarzana z polistyrenu na bazie tlenku żelaza ma doskonałe parametry użytkowe.

Budowa nowych dysków rozwiązuje też, występujący w dyskach sztywnych, problem przekłamań powodowanych wibracjami oraz osiadającymi drobinami kurzu. Zakurzenie dysków sztywnych powodowało niejednokrotnie ich niszczenie w rezultacie tarcia głowicy o powierzchnię dysku. Chociaż głowica stosowana przy współpracy z nowym dyskiem znajduje się znacznie bliżej jego powierzchni niż w dysku sztywnym, zatarcia są wykluczone dzięki odpowiedniemu wyginaniu powierzchni (bez niszczenia ścieżek zapisu).

Dyski o nowej technologii nazywane — od nazwy firmy — dyskami 3M są uznane za rewelację na rynku komputerów osobistych. Przewiduje się, że dyski 3M i przystosowane do nich napędy dyskowe skutecznie wyprą, już w 1986 roku, dotychczasowe elastyczne dyski. Spowodowane to będzie lepszą jakością oraz wielokrotnie większą pojemnością, w porównaniu do standardowego dysku elastycznego, a także stosunkowo niskim kosztem produkcji (jest o połowę tańszy od dysku sztywnego analogicznej pojemności).

Wstępne szacunki zakładają, że nowe dyski znajdą zastosowanie w 10—14 milionach komputerów osobistych, które powinny być sprzedane w latach 1985—1987. Dokładne określenie rzeczywistej ceny dysków 3M nie jest jeszcze możliwe, ze względu na nieznajomość ceny przystosowanych do nich napędów.

Opracowała TERESA WILCZEK
na podst. BUSINESS WEEKLY,

15 października 1984

Problemy tworzenia terminologii

Trudności związane z tworzeniem nowych terminów informatycznych, a zwłaszcza z ich przenoszeniem z języka angielskiego, występują, jak się okazuje, nie tylko u nas. Trudności takie odczuwają, i to w stopniu nie mniejszym niż my, także terminolodzy niemiecy. Jest to o tyle zaskakujące, że język niemiecki nie tylko należy do tej samej co angielski grupy językowej, ale również — w porównaniu z innymi językami europejskimi — ma szczególną łatwość i długoletnią tradycję tworzenia nowych terminów technicznych (głównie dzięki wyrazom wieloczłonowym — tzw. zusammengesetzte Wörter).

Trudności terminologów niemieckich potwierdza dokument roboczy nr 18 zachodnioniemieckiego towarzystwa informatycznego GMD (Gesellschaft für Mathematik und Datenverarbeitung) z października ub.r. Oprócz propozycji niemieckich terminów języka ADA („Ein Vorschlag für eine deutsche Ada-Terminologie"), dokument ten zawiera interesujące komentarze na temat powstałych w trakcie jego opracowania problemów oraz przyjętej metodyki pracy 12-osobowego zespołu autorskiego. Zapoznanie się z tymi komentarzami ilustruje istotne aspekty warsztatu terminologicznego, którego poznanie może być nie tylko interesujące, ale również pouczające zwłaszcza dla tych, którzy tak często i chętnie krytykują nasze wysiłki terminologiczne.

W rozdziale drugim wspomnianego dokumentu, zatytułowanym „Trudności" („Schwierigkeiten"), wymieniono m.in. następujące główne problemy, jakie napotykają pionierzy propozycji terminologicznych, a które wynikają z istnienia takich zjawisk jak:

- regionalne różnice w używanym już (i często mocno zakorzenionym) słownictwie; różnice te są pogłębiane przez poszczególne „szkoły", tworzące się wewnątrz większych środowisk naukowych,
- stały dylemat wyboru między propozycjami nowych terminów, a terminami potocznie używanymi, które z reguły są wieloznaczne,
- nawyki szerokiego, często bezmyślnego stosowania błędnych tłumaczeń słów amerykańskich, np. tłumaczenia angielskiego **instance** (przykład) niemieckim **Instanz** (instancja).

Głównym zadaniem zespołu autorskiego było możliwie najwierniejsze przeniesienie ustalonej angielskiej terminologii ADY na język niemiecki, przy maksymalnym uwzględnieniu obowiązujących niemieckich norm w zakresie terminologii informatycznej. Dlatego też przed podjęciem prac postanowiono przyjąć następujące reguły tłumaczenia, sprecyzowane w wytycznych Komisji Systemów Informatycznych Niemieckiego Instytutu Normalizacji (DIN — Normenausschuss Informationsverarbeitungssysteme):

- tłumaczenie powinno być komunikatywne i łatwo akceptowane przez użytkowników, dlatego należy uwzględniać słownictwo już używane w języku mówionym,
- należy stosować słowa zapożyczone z języka angielskiego i powszechnie stosowane w skali międzynarodowej (np. **bit**),
- jeśli dla terminu angielskiego istnieje podobne pod względem formy i wymowy słowo niemieckie, należy go bezwzględnie użyć, oczywiście pod warunkiem istnienia zgodności semantycznej, np. ang. **phrase** — niem. **Phrase**; ang. **option** — niem. **Option**,
- nie należy odrębnie tłumaczyć wyrażeń złożonych (wielowyrazowych) w sytuacji, gdy ich części składowe (poszczególne wyrazy) mają już ustalone terminy niemieckie,
- nie należy tłumaczyć nazw (słów) zastrzeżonych, powinny być one pisane dużymi literami.

Jak trudne jest ustalenie, a następnie uzgodnienie jednolitej terminologii narodowej, świadczą następujące dane, przytoczone w cytowanym dokumencie. Dla ok. 300 terminów angielskich wytypowanych do przetłumaczenia w momencie rozpoczęcia prac, członkowie zespołu mieli indywidualnie podać propozycje odpowiedników niemieckich. Postępowanie takie miało na celu uzyskanie rzeczywistego obrazu wewnętrznej zgodności poglądów zespołu a więc również szans szybkiego stworzenia jednolitej terminologii niemieckiej. Porównanie otrzymanych propozycji ukazało następującą sytuację: 145 (ok. 49%) słów miało tylko jedną propozycję tłumaczenia, 39 słów (ok. 13%) — dwie lub trzy propozycje, oraz 113 słów (ok. 38%) — wiele propozycji. Przykładowo, dla angielskiego terminu default zaproponowano aż 8 odpowiedników niemieckich! Dokładna analiza propozycji wykazała, że:

- słowa o charakterze ogólnym, takie jak bas, item, name, denote, null itp. nabierają precyzyjnego znaczenia tylko w określonym kontekście,
- z przetłumaczonych terminów pojęć podstawowych nie zawsze można wyprowadzić tłumaczenie terminów złożonych,
- samo przetłumaczenie terminów nie może zapewnić precyzji i spójności terminologii — oprócz tego każdy z terminów wymaga definicji i ewentualnych komentarzy,
- tłumaczenia słów o podobnym znaczeniu wymagają wzajemnego uzgodnienie.

Pierwsze spotkanie robocze zespołu wykazało konieczność liberalnego potraktowania wielowariantowych propozycji tłumaczenia terminów. Dlatego też dla wielu terminów angielskich przyjęto po kilka odpowiedników niemieckich, jeżeli było to konieczne ze względu na zależności kontekstowe (np. **base = Grund, Basis**) lub dla uzyskania większej swobody w posługiwaniu się językiem, oczywiście bez zmniejszania klarowności samego pojęcia (np. **declaration = = Vereinbarung, Deklaration**).

Podczas drugiego spotkania zespołu rozpatrzono ok. 100 propozycji zmian oraz uzupełnień słownika. Dobitnym dowodem trudności osiągnięcia kompromisu w doborze właściwych terminów niemieckich była zacytowana w dokumencie dramatyczna wypowiedź jednego z członków zespołu, który stwierdził: „Podstawowym problemem jest to, że tak często nie można znaleźć żadnego rozsądnego słowa niemieckiego!". Szczególnie trudne okazało się bowiem znalezienie kompromisu pomiędzy szowinizmem narodowym, tzn. zdecydowanym preferowaniem terminów czysto niemieckich, a bezkrytycznym przejmowaniem słów z powszechnie używanego żargonu informatycznego (w rodzaju „Ich mounte ein tape"). Sytuacja taka wystąpiła zwłaszcza w przypadku tłumaczenia takich słów, jak **array, record, string** lub **task**. Odejście od zasady jednego odpowiednika niemieckiego dla każdego terminu angielskiego nie pozwoliło niestety osiągnąć ideału rzeczywiście jednolitej terminologii. W wielu przypadkach zastosowano jako terminy specjalistyczne synonimy języka potocznego z wyraźnym jednak rozgraniczeniem ich znaczenia.

Szczególnie niejednolite były propozycje w zakresie terminów angielskich o charakterze metaforycznym, np. **parent, master, stub, box**. Przykładowo, dla terminu **parent** zaproponowano nie tylko **Vater** (ociec) i **jMutter** (matka), ale również **Erzeugung** (wytwarzanie), **Ursprung** (pochodzenie) ??? **übergeordnet** (nadrzędnie). W wyjątkowych przypadkach przyjęto terminy szeroko już rozpowszechnione, nawet wtedy, gdy po dokładnym przeanalizowaniu okazały się one merytorycznie błędne. Takim terminem jak np. **bendingte Anweisung** (instrukcja warunkowa) w odniesieniu do instrukcji IF z członem ELSE.

Zawarty w dokumencie GMD słowniczek ADY zawiera niemieckie odpowiedniki dla ok. 350 terminów angielskich a także odpowiedniki wg normy DIN 66 003 dla ok. 50 terminów dotyczących znaków specjalnych.

Terminologia

W końcowej części komentarza stwierdzono, że zawarty w dokumencie słowniczek jest wystarczający jedynie do opracowywania publikacji na temat ADY, natomiast aby zaspokoić potrzeby tłumaczenia pełnego opisu języka, wymaga on znacznego rozszerzenia (analogiczny słowniczek angielsko-francuski zawiera ok. 1000 par słów).

O stopniu trudności realizacji nawet tak skromnego zadania, jakim był omówiony projekt niemieckiego minisłowniczka ADY, świadczy fakt, że mimo udziału w pracach zespołu najlepszych w RFN specjalistów z tej stosunkowo wąskiej dziedziny, jego wykonanie trwało prawie 2 lata. Sądzę, że ta bardzo skrótowa informacja, choć w części pokazała najistotniejsze problemy prac terminologicznych, a zwłaszcza konieczność rozwiązań kompromisowych, o czym pisałem w tym miejscu w numerze lipcowym ubiegłego roku.

WŁADYSŁAW KLEPACZ

Książki nadesłane

WYDAWNICTWA NAUKOWO-TECHNICZNE

Lipowski J., Małysiak H., Pochopień B., Podsiadło P., Wróbel E.: Modułowe systemy mikrokomputerowe. Seria USE. nakład 20 000 egz., 432 str., cena 350 zł, Warszawa, 1984

W książce przedstawiono zagadnienia związane z projektowaniem sprzętowym i programowym systemów mikrokomputerowych. Projekty sprzętowe opierają się na uniwersalnych modułach funkcjonalnych, np. modułach mikroprocesorów (8080, 8085, Z80), pamięci, wejść—wyjść cyfrowych i analogowych czy programatorów pamięci. Część programowa obejmuje wybrane zagadnienia programowania systemów mikrokomputerowych w języku asemblera.

Jones C. B.: Konstruowanie oprogramowania metodą systematyczną. Seria BIO, nakład 5000 egz., 412 str., cena 330 zł, Warszawa, 1984

Przedstawiono tu metodę zapisywania projektów systemów oprogramowania, w której wykorzystuje się język wysokiego poziomu stanowiący rozwinięcie metody VDM opracowanej w Wiedeńskim Laboratorium IBM. Metoda ta pozwala na zastosowanie osiągnięć teorii informatycznej do praktycznej konstrukcji oprogramowania oraz na wykorzystanie formalnych specyfikacji do projektowania nowych systemów. Książka jest przeznaczona dla programistów, projektantów systemów przetwarzania informacji, pracowników nauki zajmujących się informatyką oraz studentów kierunków informatycznych.

Iglewski M., Madey J., Matwin S.: Pascal. Język wzorcowy — Pascal 360. Seria BIO, nakład 10 000 egz., 214 str., cena 170 zł, Warszawa, 1984

Jest to trzecie wydanie książki opisującej Pascal wzorcowy. Przy opisie języka zwrócono szczególną uwagę na niedokładności występujące w oryginalnym Raporcie języka, m.in. zmieniono prezentację pojęcia typu. Książka przedstawia też zmodyfikowaną wersję dokumentu „Pascal 360. Opis dla użytkownika". Przeznaczona jest ona dla programistów, projektantów systemów przetwarzania informacji, pracowników nauki oraz studentów kierunków informatycznych.

Karczmarczuk J.: Komputery SINCLAIRA — rachunki numeryczne w kodzie maszynowym

INFORMATYKA 1985, nr 5, s. 1

Omówienie sposobu zwiększenia efektywności obliczeń numerycznych w przypadku ich realizacji na mikrokomputerach ZX81 oraz ZX SPECTRUM.

Furman M., Nowicki Z. M., Solak J.: Pakiet SIMIS dla MERY 400

INFORMATYKA 1985, nr 6, s. 4

Charakterystyka uniwersalnego pakietu programowego do wyszukiwania informacji SIMIS, będącego rozwinięciem pakietów serii SIM-400. Omówiono możliwości wyszukiwawcze pakietu oraz plany jego rozwoju.

Winiewski J.: Język programowania C (3)

INFORMATYKA 1985, nr 6, s. 7

Dokończenie charakterystyki języka C. Omówiono mechanizmy komunikacji zewnętrznej, działanie kompilatora oraz przykładowy program.

Качмарчук И.: Компьютеры SINCLAIR — численные расчеты в коде вычислительной машины

INFORMATYKA 1985, № 5, стр. 1

Обсуждено метод повышения эффективности численных расчетов для мини-ЭВМ типа ZX 81 и ZX SPECTRUM.

Фурман М., Новицки З. М., Соляк И.: Пакет SIMIS для MERA 400

INFORMATYKA 1985, № 6, стр. 4

Характеристика универсального комплекта программного обеспечения SIMIS предназначенного для поиска информации, который является продолжением программного обеспечения серии SIM-400. Обсуждены возможности поиска информации с использованием комплекта программного обеспечения и перспективы его развития.

Виневски И.: Язык программирования С (3)

INFORMATYKA 1985, № 6, стр. 7

Завершение характеристики языка С. Обсуждены механизмы внешней связи, действие программы-компилятора и примеры программ.

Karczmarczuk J.: SINCLAIR computers — numeric computations in machine code

INFORMATYKA 1985, No. 5, p. 1

Effectiveness improving of numeric computations on ZX 81 and ZX SPECTRUM.

Furman M., Nowicki Z. M., Solak J.: SIMIS — a program package for MERA 400

INFORMATYKA 1985, No. 6, p. 4

Characteristics of SIMIS, an universal program package for information retrieval, which is developed from SIM-400 package series. Retrieval possibilities of the package and its development plans are discussed.

Winiewski J.: C programming language (3)

INFORMATYKA 1985, No. 6, p. 7

Last part of the C language characteristics. External communication mechanisms, compiler functions and an examplary program are discussed.

Karczmarczuk J.: SINCLAIR-Computer — numerische Berechnungen in Maschinencode

INFORMATYKA 1985, Nr. 5, s. 1

Eine Besprechung von Leistungssteigerung der numerischen Berechnungen im Falle ihrer Realisation auf ZX 81 — und ZX SPECTRUM-Mikrocomputern.

Furman M., Nowicki Z. M., Solak J.: SIMIS — ein Programmpaket für MERA 400

INFORMATYKA 1985, Nr. 6, S. 4

Eine Charakteristik des universellen Programmpakets für Informationsrechercheanwendungen SIMIS, das eine Weiterentwicklung von SIM-400-Programmpaketserie darstellt. Es wurden Recherchemöglichkeiten des Pakets und seine Entwicklungspläne besprochen.

Winiewski J.: C-Programmiersprache (3)

INFORMATYKA 1985, Nr. 6, S. 7

Letzter Teil einer Charakteristik von C-Sprache. Es wurden Mechanismen der Aussenkommunikation, Funktionieren des Kompilators und ein Beispielprogramm besprochen.

Konferencje

Problemy komputeryzacji projektowania urządzeń elektroenergetycznych

Międzynarodowa konferencja na temat komputeryzacji projektowania urządzeń elektroenergetycznych **odbędzie się w dniach 17—19 września 1986 roku w Szklarskiej Porębie.**

Jej zamierzeniem jest:

- przedstawienie aktualnego stanu wiedzy i techniki w zakresie komputeryzacji projektowania w elektroenergetyce,
- przedyskutowanie wpływu systemów mikrokomputerowych na przyspieszenie wdrożeń wyników prac badawczych,
- ustalenie zasad koordynacji i integracji prac wdrożeniowych,
- określenie perspektyw rozwoju metodologii i instrumentalizacji.

Tematyka konferencji dotyczyć będzie podstaw metodologicznych, algorytmizacji oraz wymagań sprzętowych i programowych komputeryzacji procesów projektowych.

W czasie jej trwania przewiduje się dyskusję poniższych zagadnień: modele procesów projektowych, reprezentacja wiedzy projektowej, bazy danych i optymalizacja w projektowaniu, sieci komputerowe, mikrokomputery i mikroprocesory w projektowaniu i modelowaniu procesów, społeczne i ekonomiczne aspekty komputeryzacji, szkolenie kadr, grafika komputerowa, przykłady algorytmów projektowych.

Referaty na konferencję można zgłaszać w terminie do dnia 15 października br. wraz ze streszczeniem referatu (od 150 do 200 słów) w językach: polskim, angielskim, niemieckim lub rosyjskim. W tym samym terminie należy zgłaszać uczestnictwo w konferencji. Koszt udział wraz z materiałami konferencyjnymi wynosi 8500 zł. Kwota ta obejmuje również zakwaterowanie i wyżywienie.

Szczegółowych informacji udziela przewodniczący Komitetu Organizacyjnego:

doc. dr hab. inż. Jan Bujko
Instytut Energoelektryki
Politechnika Wrocławska
Pl. Grunwaldzki 13
50-377 Wrocław
tel.: 20-39-60, 20-26-55, teleks: 0712254 PWr PL lub 0712559 PWr PL

ZAGADKA

Jaki jest wynik wykonania podanego programu i od czego zależy?

```
      COMMON/BLOCK/WHO(3)
      DATA A,B/4H12  ,4H  3  /
      WRITE(8,1)A,B
1     FORMAT(2A4,1X)
      BACKSPACE 8
      READ(8,2) N
2     FORMAT(A9)
      WRITE (8,1) N
      BACKSPACE 8
      READ (8,3) N
3     FORMAT (I1)
      WRITE (6,1) WHO(N)
      STOP
      END
      BLOCK DATA
      COMMON/BLOCK/CDC,ICL,IBM
      DATA IBM/4H IBM/,ICL/4H ICL/,CDC/4H CDC/
      END
```

(Odpowiedź wewnątrz numeru)

JAN BIELECKI

CSK—Computer Studio Kajkowscy

81-505 GDYNIA ORŁOWO, ul. Balladyny 3B, tel. 29-00-18

Komputer osobisty może być przydatny niemal na każdym stanowisku pracy. Wymaga jednak odpowiedniego oprogramowania użytkowego. W ramach tego oprogramowania oferujemy zainteresowanym dostawę uniwersalnych pakietów programowych:

BANK DANYCH CSK, TABPLAN CSK, TEKST CSK, TRANSCOM CSK

To doskonałe narzędzia pracy dla każdego. Aby z nich korzystać, nie trzeba być informatykiem! Zupełnie samodzielnie można tworzyć złożone systemy zarządzania przedsiębiorstwem, każdym przedsiębiorstwem; nawet najbardziej specyficzne uwarunkowania nie są przeszkodą.
To jednak jeszcze nie wszystko... Kiedy dotychczasowe problemy łatwo i szybko zostały rozwiązane — pojawiają się zupełnie nowe. Można wtedy bez kłopotów samemu udoskonalić dotychczasowy system!

BANK DANYCH CSK, TABPLAN CSK, TEKST CSK, TRANSCOM CSK

składają się w zakładowe systemy płacowe, osobowe, finansowo-księgowe lub magazynowe. Korzystając z nich, z łatwością można prowadzić planowanie, kalkulacje i sprawozdawczość. Można też sporządzać kosztorysy i oferty, a nawet prowadzić „automatyczną" korespondencję czy redagować dowolne teksty. Można wreszcie skorzystać z już zgromadzonych zasobów na komputerze ODRA (pod nadzorem systemu GEORGE-3), wykorzystując komputer osobisty jako inteligentny terminal — stację lub emulator TTY.

Nowość:

Oferujemy system operacyjny kompatybilny z CP/M 2.2. dla mikrokomputerów ROBOTRON 5120/5130 oraz systemy finansowo-księgowe FK dla dowolnych mikrokomputerów

---

Szczegółowych informacji udziela:

CSK—Computer Studio Kajkowscy

81-505 GDYNIA ORŁOWO, ul. Balladyny 3B, tel. 29-00-18

EO/129/K/85

Centralny Ośrodek Badawczo-Projektowy Budownictwa Przemysłowego

BISTYP

zaprasza do korzystania z usług ośrodka obliczeniowego wyposażonego w nowoczesny sprzęt mini- i mikrokomputerowy, autokreślarki oraz urządzenia do graficznego wprowadzania danych.

Pracownicy biur projektów przedsiębiorstw komunalnych, wydziałów architektury i budownictwa uczelni technicznych, wydziałów architektury, urbanistyki i geodezji przy radach narodowych znajdą w naszym Ośrodku następujące zautomatyzowane problemy:

- obliczenia inżynierskie w zakresie projektowania konstrukcji budowlanych, instalacji wodociągowych, kanalizacyjnych, wentylacyjnych
- obliczenia w zakresie ochrony atmosfery obejmujące rozkład przestrzenny stężenia i strumieni zanieczyszczeń powietrza emitowanych przez punktowe, liniowe i powierzchniowe źródła emisji w skali regionu
- sporządzanie planów generalnych zakładów przemysłowych.

  Obliczanie, sprawdzanie oraz rejestrowanie robót związanych z pracami ziemnymi obejmujące między innymi: bilansowanie mas ziemnych gruntu, optymalizację przewozów mas ziemnych, sporządzanie rysunków planów warstwicowych terenu i jego przekrojów, wykrywanie kolizji między projektowanymi elementami infrastruktury i budowli
- składanie i wykreślanie rzutów kondygnacji oraz automatyczne wykreślenie dowolnych przekrojów i elewacji budynków na podstawie wprowadzonych szkiców pięter i katalogu zastosowanych elementów.

Wszystkie powyższe problemy po rozwiązaniu dają wyniki przedstawione w postaci alfanumerycznej i graficznej.

Szczegółowych informacji o usługach obliczeniowych udzielają pracownicy Zakładu Nowych Metod Projektowania przy COBPBP „Bistyp", 00-518 Warszawa, ul. Parkingowa 1, tel. 28-94-71.

EO/828/K/85